Agyei Eunice Eno Yaa Frimponmaa

How to Design Activities for Learning Computational Thinking in the Context of Early Primary School in an after-school Code Club

Master's thesis
FACULTY OF EDUCATION
Learning, Education, and Technology
2019

University of Oulu
Faculty of Education
How to Design Activities for Learning Computational Thinking in the Context of Early Primary School in an after-school code club (Eunice Agyei)
Master's thesis, 72 pages, 10 appendices
April 2019

Abstract

Computational Thinking (CT) and its related concepts have gained a lot of traction within the field of education. Many countries, including Finland and the United Kingdom, are in the process of integrating CT into their national curriculums to equip pupils with much needed 21st century digital skills, including coding (programming). As a result, several programs and activities are being developed to introduce pupils to CT. The need to develop appropriate teaching and learning materials, as well as train teachers to teach, and integrate computational thinking into their lessons is apparent. This thesis seeks to contribute to the body of knowledge on computational thinking by designing and testing instructional materials for early primary school. Computational thinking as a concept, how to integrate its concepts into coding, as well as how pupils understood the concept were explored.

This study was conducted in an after-school coding club at an elementary school in the northern part of Finland. The duration for the coding club was 8 weeks. Each lesson lasted for 45 minutes. Participants were selected from among 1st and 2nd grade pupils. In selecting participants for this study, priority was given to pupils with no prior coding experience. 13 out of the selected 17 had no prior experience. The remaining 4 participants were randomly selected from the rest of the applicants who had coding experience.

Worksheets and stickers were designed and tested for teaching and learning computational thinking. Lesson plans designed for the coding club included activities for teaching computational thinking using unplugged activities and Scratchjr. The unplugged activities were integrated into coding lessons to enhance the understanding of pupils during the coding lessons. This approach helped to connect theoretical computational thinking to real life practices and its application in the context of coding.

Data collected included the unplugged activity worksheets of the participants, their Scratchjr projects, and self-efficacy beliefs regarding their ability to code and think computationally. These work products were evaluated qualitatively for evidence of understanding. The analysis of the self-efficacy beliefs of participants revealed that participants were confident of their computational thinking and coding abilities.

The main outcome of this research is the instructional material (stickers, templates, and Scratchjr activities) which was designed for teaching and learning purposes. This unique experiment and pedagogical designs are explained to show how unplugged activities can be used to introduce pupils to computational thinking concepts.

Keywords: computational thinking, instructional design, coding, unplugged, worked example

FOREWORD

I thank the Lord God Almighty who has been the author of my life. I am grateful for all that He has done for me and how far He has brought me. Secondly, I thank my supervisor, Dr. Jari Laru for his guidance and direction. This thesis would not have been successful without him. I also thank Mr Jani Vallirinne and Ms Sirkku Tahvanainen of Code School Finland for the thesis opportunity.

My gratitude goes to my parents Mr. Kwaku Adiefeh Agyei and Mrs. Matilda Adjei-Sarpong for their love, support and sacrifices they have made to bring me this far. A thank you also goes to sister, Christiana Akua Akyaa Agyei, and my cousin Nana Adjoa Asantewaa Asiamah for been an inspiration to me. To Mr. Ebenezer Adjei-Sarpong, Mr. Enoch Adjei-Sarpong, Ms. Adjoa Addai, Mrs. Bridget Adoma, Ms. Claudia Hilton Aba Hamilton, Mrs Kate Asiamah and the families of Adjei-Sarpong, Adiefeh, and Asiamah, I say a big thank you. I will not forget all your support.

I would like to thank my class mates and friends Mr. Opoku Asare Kennedy, Mr. Mutai Kenneth, Miss Xinru Chen, Mr. Ahmad Ghaznawii, Mr. Hassan Mehmood, Mr. Dat Le Quoc, Mr. Bram-Larbi Yaw and all who were there to support and encourage me one way or the other. A special thank you to the small Ghanaian community at the University of Oulu. Cheers!

To these special people in my life: Mr. Janne Rasanen, Mr. Andy Alorwu, Mr. Solomon Mensah, Mr. Alexander Ofori Boateng, and Mr. Christian Baidoo, I say thank you. Indeed, you are men who stand behind women. Your support is invaluable to me and I am lucky to have you all in my life. Thank you so much.

Contents

Figures

Tables

1 Introduction

In the past couple of years, Computational Thinking (CT) and its related concepts have received a lot of traction within the field of education. Many countries including Finland and the United Kingdom are in the process of integrating CT into their academic curricular to equip pupils with much needed 21st century digital skills. As a result, several programs and activities are been developed to introduce pupils to CT (Bocconi, Chioccariello, Dettori, Ferrari, & Engelhardt, 2016). According to Wing (2008), CT is a basic skill for everyone with problem solving, systems design, and understanding human behavior as its center.

CT draws on a wide range of cognitive concepts such as abstraction, decomposition, pattern recognition, and algorithms to assess and identify a problem, and design solutions to the problem (Wing, 2008). Wing (2008) expressed the opinion that CT should be taught just as reading and writing are taught. It is interesting to note that CT has been integrated into many curricular around the world through subjects such as mathematics, informatics, and information technology (Bocconi et al., 2016) but not as a standalone subject. In Finland, CT is taught as part of Math and Crafts (Mannila, 2014). Learning CT occurs through formal education, informal education, self-directed learning, or community-based learning. It can be deduced that learning occurs through individual, co-operation, and collaborative efforts.

Wing (2011) defined CT as a thought process that involves formulating, designing solutions to problems, and representing the solution in a form suitable for information processing. CT skills are acquired through understanding problem description, decomposing the problem, collecting, analyzing, and representing data, solving the problem at hand, recognizing patterns, developing algorithms, decomposition of problems, abstract reasoning, questioning, designing and simulating (Wang & Wang, 2016; Wing, 2008).

Learning of CT can be frustrating task for pupils. However, CT empowers pupils to be confident, persistent, open minded, and a team player. It also enhances their problem resolution and communication skills (Barr, Harrison, & Conery, 2011). CT represents an umbrella of knowledge, skills and tools to automate and speed up the everyday life problem-solving approach (Barr et al., 2011).

Programming is one of the 21st century skills that should be taught in schools around the globe. The national core curriculum in Finland aims that all pupils should be exposed to some form of computer programming during their primary school studies. Kazimoglu, Kiernan, Bacon, &

Mackinnon (2012) have argued that computational thinking is a stepping stone to prepare pupils to code. However, in order to increase the role of computational thinking in the primary schools, appropriate instructional materials, pedagogical designs and in-service teacher education are needed. In this thesis, instructional materials and pedagogical design are created and tested in the context of an after-school K-2 code school to address that need.

The rest of this thesis is organized as follows. After the introduction part follows theoretical framework where computational thinking and its practical applications are presented. Aim and research questions are presented after the theoretical framework followed by the methods chapter. The methods chapter describes the design of the after-school code club experiment and instructional materials. It also describes data collection and analysis methods in details. The results, discussion and conclusions are presented in subsequent chapters.

2 Theoretical framework

2.1 Computational Thinking

According to Tedre & Denning (2016), the current definition of computational thinking, was birthed from three (3) historical trends that occurred between the 1950s and 1990s. Descriptions, claims, arguments, narratives, and debates characterized this era when scientists and researchers explored computing in many forms. From the 1950's, many of the descriptions centered on algorithm thinking until the 1980s when a non-computing scientist with focus on both artificial and natural information processing pioneered a revolution. In 1980, Seymour Papert published Mindstorm in which computing ideas, techniques, technology, and language played a huge role in knowledge construction and speculated how computing can activate learning, thinking, cognitive and emotional wellbeing (Papert, 1980). His idea of constructionism occurs through exploration and constant practice. This stands out in terms of educational pedagogy although other educationists advocated and initiated computing literacy and programming (Tedre & Denning, 2016). The advocacy of scientists in the 1980s lead to a political movement and the passing of a law, the High-Performance Act which provided funding for solving the grand challenge problems identified in computing science. A key issue was the formulation of algorithmic solution to the grand challenges that will run on supercomputers (Tedre & Denning, 2016). From these historical trends, Tedre and Denning (2016) defined computational thinking as "the habits of mind that many of us have developed from designing programs, software packages, and computations performed by machines - offers very powerful mental tools for people who design computations''.

In 2006, computational thinking gained popularity in the context of education when Jeannette Wing wrote about it in her seminal essay. Her descriptions of computational thinking in no specific order includes algorithms, automation, abstraction, analytical thinking, understanding human behavior, designing systems, and mathematical thinking (Wing, 2006, 2008). She made the following assertions:

1. Computing is about automations and abstractions.
2. Computing is in many disciplines. Based on this, she suggests the use of computing will lead innovations in many fields in the future.

3. Computing is everywhere and part of our daily lives. If computing directly or indirectly affects our lives in many ways, then there is the need for everyone to acquire computing knowledge at least. She envisions computational thinking to be inherent in education especially early childhood. She states that science, technology and society are the key drivers of computing.

4. She argued that computing is more than just programming, computational thinking is a way humans solve problems instead of a way in which computers think and combines both mathematical and engineering thinking.

In 2010, Wing with other colleagues defined computational thinking as the thought processes in problem and solution formulation. "Computational Thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (Cuny, Snyder, & Wing, 2010).

The International Society for Technology in Education (ISTE), Computer Science Teachers' Association (CSTA) together with Higher Education Leaders, K-12 education, and the industry experts developed an operational definition for ICT (ISTE & CSTA, 2011). They defined CT as a problem-solving process that includes:

- Formulating problems in a way that computational tools can be used to solve.
- Organizing and analyzing data.
- Data representation using different abstractions like simulations, and modeling.
- Algorithm thinking for automations.
- Identifying and evaluating possible solutions, selecting the best solution, and implementing the solution effectively and efficiently.

In addition, they identified behaviors that CT develops and enhances. These include confidence, persistence, tolerance, ability to handle open ended problems, and ability to communicate and work in a team to accomplish the task at hand.

In summary, it appears that the definitions and descriptions of computational thinking by Wing and the operational definition offered by the ISTE and CSTA corresponds with the definition that arose from the history overview of Tedre and Denning.

Several computational thinking activities allow pupils to learn through collaboration, cooperation and self-direction. CT focuses on problem solving, operational understanding of CT concepts. It requires no prior knowledge and more importantly no computers are required hence pupils everywhere in the world can easily acquire CT knowledge.

The learning of Computational Thinking in the context of coding can be categorized into plugged and unplugged activities. (Atmatzidou & Demetriadis, 2014). Unplugged activities typically mean teaching or learning computer programming and science without a computer (Code.org, 2013). Unplugged refers to kinesthetic activities that teach computational thinking (Code.org, 2013) without the use of computers. It focuses on understanding and applying algorithms that make computers work as they are designed. Knowledge of these algorithms empowers pupils to develop coding skills. The term "plugged" describes teaching and learning of CT using computing devices**.** Block based programming, educational robots, teaching scripts, games and animations, and computer simulation are ways of teaching and learning computational thinking.

Teaching and learning of computational thinking without the use of the computers (coding unplugged)

Unplugged activities present a way of teaching CT with no or limited access to computers (Nishida et al., 2009). They provide the opportunity for pupils with no prior knowledge and skills to explore CT in simple and sophisticated ways. Unplugged activities are not restricted to any location. The activities take place in physical spaces (Nishida et al., 2009). Their kinesthetic nature enables pupils to move and learn at the same time hence it is suitable for pupils of all ages especially primary schoolchildren (Namukasa et al., n.d.; Nishida et al., 2009).

Pupils gain inspiration and interest in computer science activities by engaging in non-screen activities which could build up their knowledge in the understanding of foundational concepts of computing, giving tangibility and visibility to abstract concepts within the field of computer science (Curzon & Paul, 2013; Thies & Vahrenhold, 2013). Due to its collaborative nature, unplugged activities provide an additional benefit of shifting pupils' understanding about what computer science is, thus children can be introduced to computer science and its related subjects as a means of enhancing their problem-solving skills (Curzon, McOwan, Plant, & Meagher, 2014).

The adoption and use of unplugged activities provide opportunities for everyone to learn especially those without access to computers or are too young to use them well. According to Lamagna (2015), novices and young pupils are freed from implementation details that are required for the running of a computer program with unplugged activities. This encourages active participation as there is limited interaction with the computer screen (LeMay, Costantino, O'Connor, & ContePitcher, 2014).

Indeed, even though programming is an indispensable skill, unplugged activities allow pupils to visualize and witness the process of completing a task while contextualizing computing (Curzon et al., 2014). While computing concepts are made simpler and easier for pupils to understand, it also increases their motivation and interest as they find joy in working collaboratively.

According to Lambert (2009), using unplugged activities, fourth grade pupils had an improvement in their interest in computer science, confidence in mathematics, as well as cognitive skills.

Table 1 Describes resources that can be used for teaching and learning computational thinking. Code.org, CS Unplugged, Scratchjr have curriculums that aid teachers to deliver coding lessons

Name	Description	Level/Age	Website
CS Unplugged	A website with a collection of teaching and learning materials for teaching computer science without computers using games and puzzles.	K-4 (4-9)	https://csunplugged.org/en/
Scratch	Scratch is a programming language and a platform designed to teach programming via interactive stories, games, and animations. It was developed by the Lifelong Kindergarten Group at the MIT Media Lab. Scratch was designed to give room for pupils to tinker, create games, interactive stories and art, and share their projects (Resnick et al., 2009). Scratchjr provides opportunities for younger pupils to learn programming using preprogrammed blocks.	K(4-17) - college	https://scratch.mit.edu/
Scratchjr	Scratchjr provides opportunities for younger pupils to learn programming using simple preprogrammed blocks.	K-2(4-8)	https://www.scratchjr.org/
Code.org	Code.org is an organization that offers coding courses and resources for teaching and learning computer science. They also offer hour of code lessons for schools. They have resources for both	K-12(4-17)	https://code.org/

	plugged and unplugged approaches for learning computer science.		
Alice	Alice is a free to use app developed by the Stage 3 Research group at the Carnegie Mellon University in 1995. Originally, it was developed as a Virtual Reality (VR) prototyping tool for non-programmers to share VR content. Alice consists of 3-D model objects and instructions, which are in the form of scripts.	Varies	https://www.alice.org/
Light Bot	Lightbot is a platform designed for novices to learn programming through puzzles. By navigating a robot via a maze, programming concepts such as sequencing, procedures, loops, and conditionals are explained to pupils. The pupils are given a challenge in which they apply the concepts they have learned. No prior knowledge of programming is needed.	Pre K- K-2 (3-8)	http://lightbot.com/

Block based coding (computer programming)

Computer programming is the process of designing, developing and implementing instructions that directs a computer to perform certain tasks. Computer programming is implemented using a specific programming language (Balanskat, & Engelhardt, 2015). Writing programs to control a computer requires expert knowledge of the context in which the application will be used, algorithms and logic. Analyzing the context of the problem leads to an in depth understanding of requirements and initiates a search for solutions. Requirements of the solutions are identified and verified. Implementation of the solution identified occurs by writing code in a target language. In this phase, programming constructs such as variables, algorithms, control, and loops are used (Maloney, Peppler, Kafai, Resnick, & Rusk, 2008).

Programming is the process of analyzing a problem, designing a solution and implementing the solution. Coding is a phase in programming, thus implementation. It includes implementing the solution in a specific programming language, testing and debugging (Duncan, Bell, & Tanimoto, 2014). Computational thinking is an abstract term that describes thought processes or ways of thinking when using or designing computing systems (Wing, 2006; Bocconi et.al, 2016). In this paper, coding and programming are used synonymously.

Using computer Simulation and Modelling as a tool to teach or learn computational thinking

Using computer simulation and modelling tools is a way to teach or learn CT. Pupils learn to design and implement models and run simulations of the models created (Moursund, 2015) "Growing up Thinking Scientifically" project is an example of that (Lee et al., 2011). In this project, the pupils used abstraction to narrow down a real-world problem, followed by creating a model with the features they selected from the real-world problem. A typical model was created to predict how disease will spread in a school. The school population, layout, number and movement of pupils were among the features selected for this model. The model constructed was tested with a robot and then evaluated to know how it reflected the real-world problem (Lee et.al, 2011).

Game design as a method to learn computational thinking skills

Game design is another method for pupils to learn computational thinking skills. Pupils who participated in the iGame program, created their own game through story telling by developing scenes using Alice, which is a 3D animation program (Lee et al., 2011). Games can be used to learn computational thinking concepts and practices, like algorithmic thinking, programming, modeling, and abstraction (Werner, Denner, Bliesner, & Rex, 2009).

In addition, Scratch, block-based programming environment, can be used for creating animated stories and games. Yet, Minecraft is a game that allows pupils to design their own artefacts during the gameplay by using programming tools integrated in the game. In creating such artefacts, pupils can acquire computational thinking concepts and learn different computational thinking practices.

Using educational robots objects or tools for learning

Educational Robots can be used as objects of learning and as a tool for learning. When a robot is an object of learning, the pupil constructs a robot. On the other hand, when a robot is a learning tool, it can be used to teach subjects such as Mathematics, Science, and computational thinking (Burbaitė, Damaševičius, & Štuikys, 2013). For example, robots were used by Denis and Hubert (2001) for teaching computational thinking via a collaborative problem-solving approach. The pupils were given roles such as analyst, algorithm designer, programmer and debugger, which were alternated during each activity. In their research, they discovered that pupils understood and adopted CT concepts as the projects progressed and recommended more authentic and engaging problems as it enables pupils to grasp CT concepts.

2.2 Assessment and Computational Thinking

2.2.1 How to asses computational thinking in the context of Scratch coding platform: Brennan's and Resnick's model

Brennan and Resnick conducted a research on how to assess CT on Scratch. They came up with 6 factors to consider when assessing CT (Brennan, & Resnick, 2012). These include: 1) *Meaningful and pupil centered assessment*: Assessment must support pupil's interest and goals by providing self-exploratory pathways, multiple design scenarios for pupils according to their strength and abilities, and opportunities for reflection; 2) *Assessing artefacts*: Collecting and evaluating the pupils' projects and products reveal the challenges of the pupil. With that information, the needed guide designed for the pupil; 3) *Document learning process*: Encourage pupils to take notes, videos, audio, and record the screen of their working process to present, discuss with peers and reflect later. The purpose is to develop metacognition towards acquiring the ability to regulate their own learning; 4) *Continuous assessment*: To appreciate how far the pupil has come, and guide him or her, knowing the starting point, current level, and their goals. Taking multiple point assessments, gives snapshots of pupils' CT experience, progress and challenges. These snapshots are potential sources of relevant information for support purposes; 5) *Assessing multiple ways of knowing*: Knowledge of concepts and practice are both relevant in assessing pupils. Knowledge of concepts accounts for the ability to list, identify, define, explain, and give examples of CT concepts. Knowledge of practice on the other hand, shows how the concept is used in practice. For instance, knowledge of repetition (CT practice) in the form of loops in addition with knowledge of pattern recognition (a CT concept); 6) *Assess with multiple viewpoint*s: With the success of the pupil in mind, the viewpoint of the pupil, peers, teachers and if possible, soliciting for parents', teachers', experts', and/or researchers' opinions of pupil's work and ways to support the pupil. Parents' opinion may be valuable if they know how well their children are progressing.

2.2.2 Using Bloom's Taxonomy for assessing computational thinking

Blooms Taxonomy is another way to assess pupils (Fuller et al., 2007; Starr, Manaris, & Stalvey, 2008; Thies & Vahrenhold, 2012). This taxonomy was also used by (Starr et al., 2008; Thies & Vahrenhold, 2012) to assess CT concepts in tasks, projects, and products. Blooms Taxonomy is made of the cognitive, affective and psychometric domains (Bloom, 1956). Fuller

et al. (2007) and Starr et al., (2008) agree to the use of Bloom's Taxonomy for eliciting learning objectives and assessing pupils. The research by Starr and his colleagues spanning over a 3-year period highlights the need to use the taxonomy for setting learning objectives because it guides the selection of pedagogical tools, strategies, and suitable tasks for CT lessons. According to Starr et al. (2008), the Bloom's taxonomy represents the pathway for mastery of a CT skill as it can be used to estimate the time and effort required to teach CT knowledge and skills. By outlining the stages of knowledge acquisition, teachers will know the level pupils are and provide the needed support.

2.2.3 Assessing computational thinking by using formative and summative assessments

CT can also be assessed using formative and summative assessments (Yadav et al., 2015). In their research, Yadav and colleagues interviewed CT teachers. These teachers used formative assessment to monitor the progress of their pupils. The teachers collected feedback from pupils, and this informed decisions regarding future instructions. Summative assessment was used to assess mastery of CT concepts and products created by the pupils. This form of assessment often results in grades, which 'describes pupils' mastery'. Multiple Choice Questions (MCQs), open-ended questions, and rubric, were used for summative assessment. While MCQs and open-ended questions were used to solicit pupils understanding of CT concepts, rubric were used for setting expectations which pupils had to accomplish if they desired a grade. This gives pupils the opportunity to make important decisions on how much effort and time they are willing to invest to attain a grade or goal.

3 Aim and Research Questions

The purpose of this research is to design, use and evaluate pedagogical practices and materials for teaching CT and coding to elementary school pupils. This research aim can be further divided into three main questions:

RQ1: How were unplugged and plugged coding activities designed in order to introduce computational thinking concepts to pupils?

RQ2: How was pupils' understanding of basic computational thinking concepts visible in the artefacts that they did produce during the coding club?

RQ3: What differences exist in the self-efficacy beliefs of learners regarding computational thinking and coding abilities with respect to prior experienced and gender?

4 Methods

A major part of this thesis is related to the design of instructional material and pedagogical activities for unplugged and plugged coding activities in the context of an after-school coding club. The processes employed were influenced by Design Based Research methodology. It is a systematic methodology that seeks to bridge the gap between theory and practice through iterative analysis, design, development, and implementation, emphasizing on collaboration between researchers and practitioners (Wang & Hannafin, 2005). DBR provides insights on improving teaching and learning using existing theories (Barab & Squire, 2004). DBL is a cyclical methodology with phases and evolves through iterations.

According to EduTech Wiki by University of Geneva (2019), Design-based research can be seen as a combination of action research and ordinary instructional methods such as ADDIE. In this study, the author designed her instructional materials and pedagogical design based on the feedback that she got during the intervention in the meetings of the coding club. However, the design of the code club was based on an established instructional design model (ADDIE). See Figure 1 to see how the ADDIE model was used in this thesis.

ADDIE is an instructional design methodology that consists of 5 phases; analysis, design, development, implementation, and evaluation phases for creating and developing instructional materials and learning objects that impacts knowledge and skills (Clark, 2011).

- *Analysis phase:* includes studying the context to understand it, identifying the problem, existing knowledge gaps, constraints, and formulating goals and objectives to address the identified needs.
- *Design phase*: includes elaborating learning objectives, sequencing the learning objectives, creating learning scenarios for each objective, and outlining the learning materials, tools, materials, and equipment needed.
- *Development:* This phase includes producing the instructional or learning materials.
- *Implementation:* This phase includes conducting teaching experiments or lessons to test learning materials.
- *Evaluation:* This phase involves eliciting tools to measure and monitor the effectiveness of the intervention. Summative and formative methods are used to assess whether the learning materials achieved set objectives.

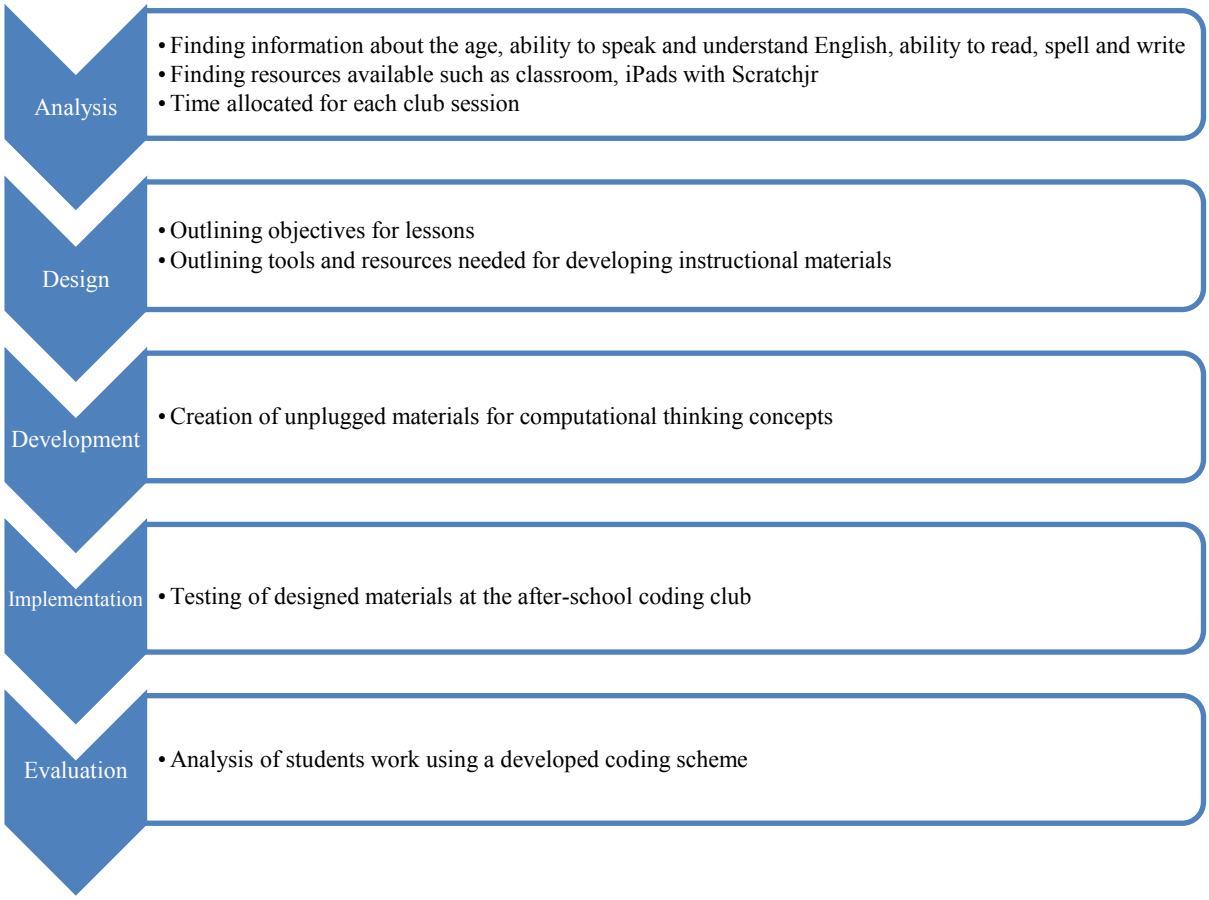


Figure 1 Description of how ADDIE model was used in this thesis

4.1 Participants and Context

The context of this study was an after-school coding club in a primary school in the northern part of Finland. A total of 17 children (8 girls and 9 boys aged 6-8) were selected to participate in the lessons organized in the coding club. Sampling of the participants was based on their previous coding experience (no experience / any experience). However, only 13 out of the selected 17 fulfilled the criterion with the other 4 participants randomly selected from the rest of the applicants who had coding experience.

Code club meetings were organized into 8 lessons, where each lesson lasted for 45 minutes. The author of the study was the designer of the activities and materials. She was also the responsible teacher in the club, although she got help from participants' parents and other stakeholders.

The sessions were arranged in a school computer laboratory. Each participant received an iPad labelled with their names that they used to do the required plugged coding exercises in the

context of the club. Scratchjr was used as the coding tool (see Figure 8) to teach computational thinking in the context of the plugged coding part of the design.

4.2 Pedagogical Design

4.2.1 Design Principles which were used to guide instructional design and pedagogical activities

Learning theories and methodologies influenced the design of the coding and unplugged materials. Reigeluth (2013) suggests theories and methodologies for sequencing and delivering lessons. The basics of the design included three major objectives: 1) *Learning objectives guides the planning and delivery of lessons.* It also guides the assessment of pupil and instructional material. It defines the topics and concepts to be covered as well as the level of understanding for them hence its suitability for assessing learning; 2) In designing coding instructions, *worked examples* were used to facilitate the process of knowledge transfer thus to the reduce cognitive effort needed to master CT and coding concepts; 3) *Assessment is a tool to identify and measure the comprehension of the pupils*. Pupils often have different attitudes towards learning; deep and surface learning (Harlen & James, 1997). While deep learning pupils seek to understand and obtain mastery, surface learning pupils seek to reproduce content, get good grades as an indicator of performance. Understanding concepts enables pupils to use and reproduce the knowledge acquired in other contexts (Harlen & James, 1997). In the context of this study, the author made use of self-assessment during the entire 8 weeks to effect changes in the instructional materials and pedagogical materials when needed. This reflection can be seen from the personal diary (Appendix II).

Instructional design and pedagogical activities were also influenced by concepts like: 1) *Authentic learning*: This type of learning supports learning by providing real life examples that pupils can relate to; 2) *Direct instruction*: This approach was for scripting both coding and unplugged activities instruction; 3) *Hands-on-learning*: It provides opportunities for exploring, discovering and experiencing concepts; 4) *Scaffolding*: This approach was used to support and facilitate pupils who needed help to understand and reach their stage of Zone of Proximal Development; 5) *Personalization*: Customizing opportunities within the instructional materials to accommodate the personal needs of pupils. Personalization was achieved by providing gender specific stickers and accommodating the different routines of pupils. For example, while some

pupils dress up before eating, some dress up after eating; 6) *Worked example:* Solved examples with steps reduces the cognitive load used for knowledge and skill acquisition and improve learning as a result.

4.2.2 General overview of the instructional design

Pupils had the opportunity to use the concepts they have learned in previous lessons. For example, the decomposition unplugged lesson followed the algorithm lesson. During the decomposition lesson, the worksheet was designed to breakdown the weekly activity performed by pupils according to the days of the week. Within this lesson, the concept of algorithms was reinforced by creating an algorithm of daily activities for each day of the week. Through this lesson, pupils learned to do some activities that repeat such as waking up, eating, and sleeping.

The plugged version of this lesson made use of repeat blocks in Scratchjr to teach the concept of repetition while applying the start and end blocks used in the algorithm (sequence). Pattern recognition is the follow up lesson where pupils identified similar patterns within their routines. With this incremental learning approach, the pupils would master the concepts despite the limited time available.

The 7th and 8th lessons were designed purposely to assess the CT and coding knowledge of pupils. One task was to write a story. The pupils were given prompts to guide them to write animated stories. A Scratchjr program was run and pupils had to write the code. This is an assessment prescribed by the creators of Scratchjr. Assessment for learning was used to evaluate pupil's comprehension of CT and coding concepts. This was achieved by evaluating pupil's work products for evidences of understanding. The success criteria for the instructional materials was the ability of pupils to apply the concepts learned to create a Scratchjr program. Self-efficacy questionnaires were used to understand the perceived confidence of learned regarding their knowledge of CT concepts and coding.

4.2.3 Lesson plans for unplugged and plugged coding activities

Using the information obtained after analyzing the context and participants, preliminary lesson plans were drafted for the plugged and unplugged lessons (See Tables 2 and 3). The main target for lesson plans was to 1) integrate computational thinking concepts into the design of coding lessons; 2) design authentic lessons where pupils would be engaged with unplugged coding

tasks (basics of the CT connected with real-life examples) and then 3) smoothly introduce participants to plugged coding tasks.

In the coding club design, the *unplugged part of the lesson f*eatured one CT concept. The CT concept was taught during the first 15 minutes. A simple definition of the concept is given followed by an explanation of the concept using storyboards, and stickers. The stickers and worksheet were packaged for each pupil due to the time constraints. The pupils were expected to put the stickers on the storyboard using glue tag given by the teacher. An alternative approach would have been to allow the pupils to cut the stickers themselves through which they would have acquired handicraft skills. However, due to the time constraints, the stickers were pre-cut.

Table 2 Unplugged Lesson Plan with description of activities and Computational Thinking concepts

Week	Description	Unplugged activity	CT concept
1	Pupils are introduced to binary code as the language computers understand. Pupils were given binary code for the alphabet and are asked to write their names in computer language.	Introduction to computing language with binary coding activity.	
2	Pupils are introduced to algorithms using their daily routines.	Introduction to algorithms using algorithm worksheet	Algorithms
3	Pupils are introduced to decomposition using their daily routine for a week. Pupils are asked to break down their routine for the week based on the day. They are also asked to identify activities they do every day and those they do only on some days.	Introduction to decomposition and pattern recognition using decomposition and algorithm worksheets.	Decomposition and pattern recognition
4	Pupils were introduced to decision making by organizing activities and clothes based on the season.	Introduction to decisions making using a worksheet	Decision making

5	Learning to include (add) or exclude(remove) activities to and from their daily routines.	Introduction to abstraction using a worksheet that simplifies the process.	Abstraction

The plugged coding lesson part further explained the *CT* concept by applying it through Scratchjr exercises. For example, in the unplugged lessons, a pupil can create an algorithm of their daily routines. A simple example is the flow of day: from the first thing done in the morning to the last thing done before sleeping. The principle behind this kind of algorithm lesson is sequencing. The understanding of the sequencing concept can be further elaborated as a coding concept by teaching them how to use Scratchjr to create code. In practice, they were shown how to use a start and an end point blocks in their Scratchjr code and how to put other blocks of the code in between them logically.

Table 3 Plugged Lesson Plan with Scratchjr describing activities and CT concepts

Week	Description	Plugged activity with Scratchjr	CT Concept
1	Introduction to Scratchjr interface	Introduction to Scratchjr	
2	Learning about sequences using motion blocks Using start and end blocks to start and end Scratchjr code	Introduction to algorithms using sequences	Algorithms and sequences
3	Learning repetition using the numbers on Scratchjr blocks, repeat block and forever block Continue to learn how to use the repeat and forever block	Introduction to controls	Repetition
4	Learning to use color coded message blocks to send and receive messages to and from Scratchjr characters.		Decisions
5	Learning to modify Scratchjr characters and background, add sound blocks, record sounds, and modify characters with pictures taken with camera. In this lesson, pupils include	Introduction to abstractions	Abstraction

	and exclude by modifying some default features with Scratchjr. Also, pupils get to decide the blocks important for their code and those that are not.		
6	Revising learned CT concepts. Pupils reviewed the concepts by writing Scratchjr code that demonstrates concepts. This prepared them to create their own Scratchjr projects.	Revision of Concepts	Algorithms, decomposition, decisions, and abstraction
7	Creating a story or game using the concepts that have been taught.	Create your own project	Algorithms, decomposition, decisions, and abstraction
8	Continuation of project and wrapping up. Taking self-efficacy survey to evaluate one's beliefs		Algorithms, decomposition, decisions, and abstraction

Table 4 Integrating computational thinking concepts between unplugged and plugged (coding) activities

CT concept	Unplugged Activity	Coding Activity	Simple Explanation of concept	Rational of concept from literature
Algorithms	Write an algorithm for daily routine	Write a Scratchjr code to move a character from one point to another	Sequence of activities with start and finish end points	"designing a step-by-step solution to a problem" (Mannila, et. al 2014)
Decomposition and Pattern recognition	Write an algorithm for weekly routine	Write a Scratchjr code for a character to do something repeatedly	Decomposition: Break down given the problem to identify the basic components. Pattern recognition: Identify actions that occur more than once.	"formulating a solution to a larger problem and breaking the solution down into smaller tasks to be dealt with" (Mannila, et. al 2014) Pattern recognition "is observing patterns, trends, and regularities in data" (Mannila, et. al 2014)
Decision making	Write an algorithm to do something	Write a Scratchjr code to control the	Presence of start and end blocks. Use of control blocks	Using cognitive skills or strategies to evaluate out-

	based on the season	movement of a character	for sending and receiving message using color coded start blocks	comes of our thought processes (Halpern, 1998)
Abstraction	Write an algorithm to show important routines in a day	Write a Scratchjr code to include important blocks, background, or character	Simplifying things by hiding some details (adding important things and removing unimportant things)	"is identifying and extracting relevant information to define main idea(s)" (Computational Thinking Concept Guide, n.d)

4.3 Instructional materials and tools

4.3.1 Activity worksheets for unplugged coding activities

Stickers for unplugged coding with activity worksheets

This sheet (Figure 2) shows sample stickers used by pupils while learning concepts of computational thinking using activity worksheets. All examples below are from the situation where

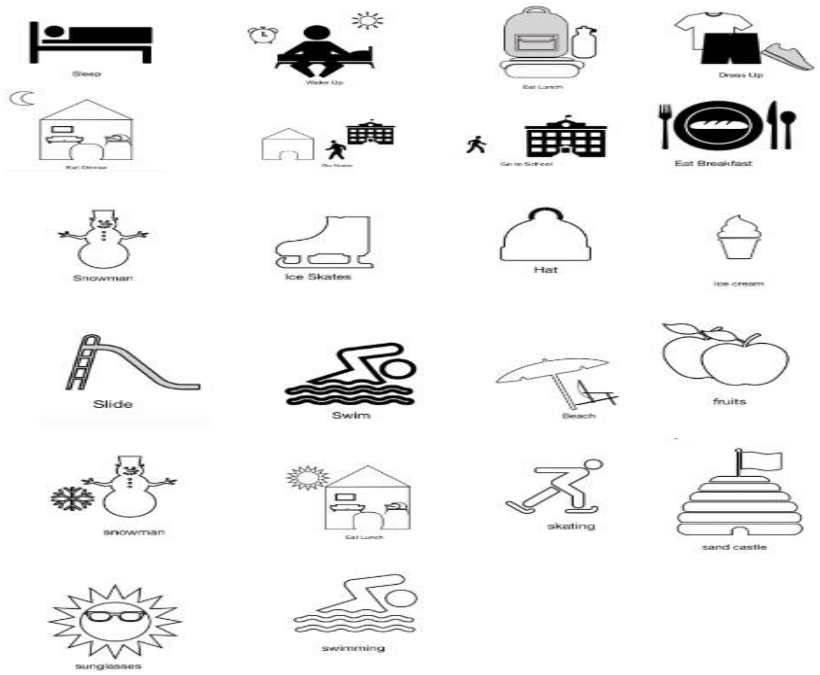


Figure 2 Stickers for unplugged activities

stickers are used with the (initially) empty forms. Stickers can be customized to fit into each learning situation and need.

Algorithm worksheet – recognizing daily routines

The purpose of the worksheet is to help in the teaching and learning of algorithms through daily routines. The pupils are required to use stickers to outline their daily routine from the very first thing they do in the morning to the last thing done in the evening. The numbers in the worksheet guides the pupils to organize their daily routine in an orderly sequence. See Figure 3.

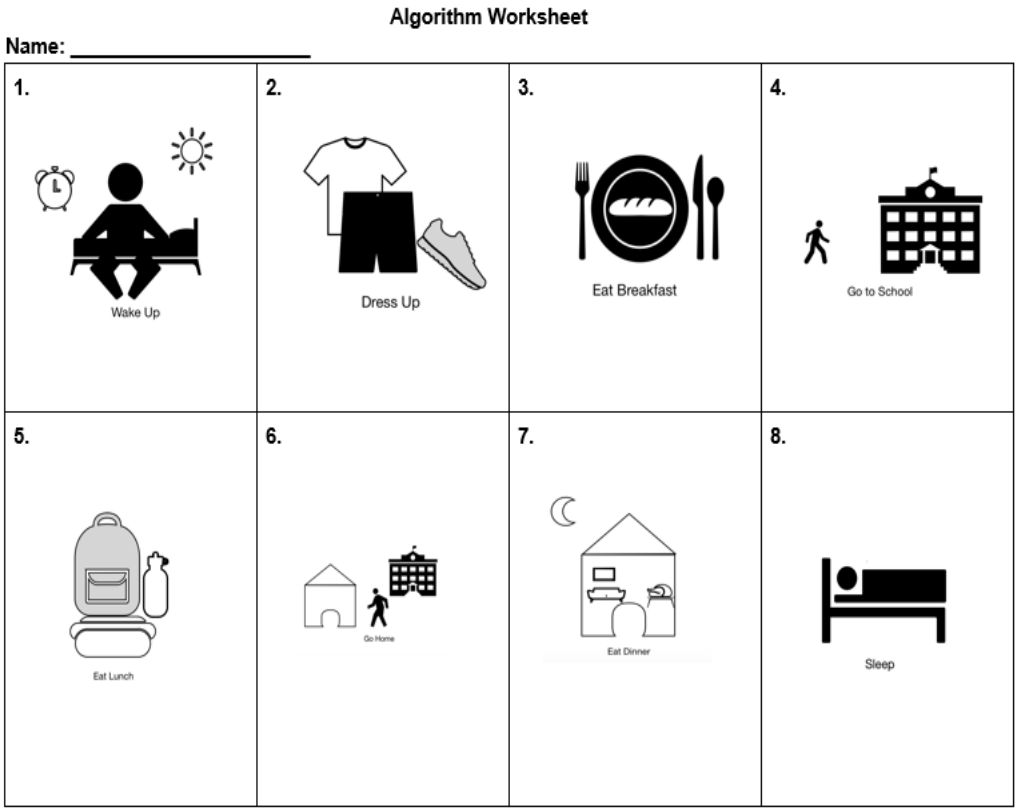


Figure 3 Algorithm worksheet describing the daily routine of pupils as an algorithm

Decomposition Worksheet – weekly routines were broken down by week days

This worksheet supported the teaching of decomposition. See Figure 4. The weekly routine of the pupils was broken down by the days of the week. The pupils were required to outline their routines for each of the days in an orderly manner. This worksheet also enforced the application

of algorithms from the previous worksheet. Pupils identified repeated routines by themselves

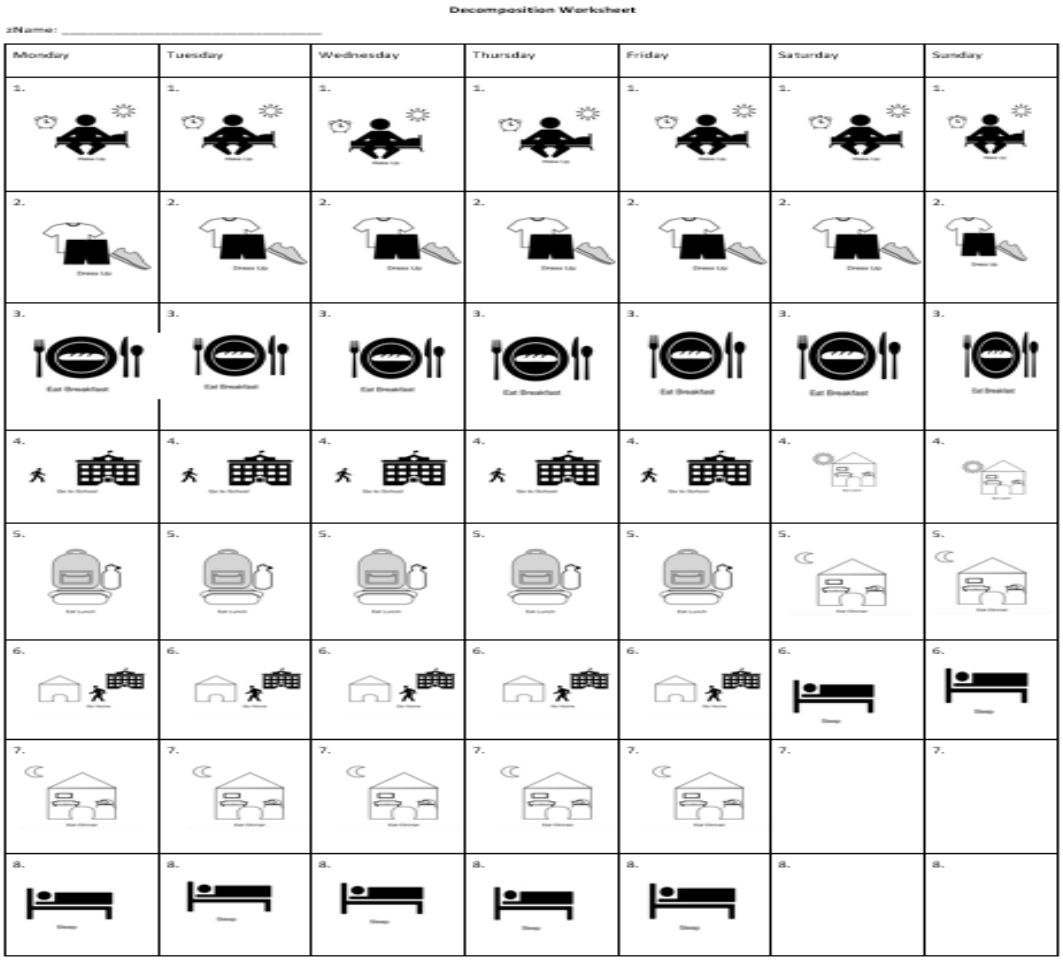


Figure 4 Decomposition worksheet to break weekly routine of leaners by the day of the week

or are prompted to do so by the teacher. The numbers again guided the pupils to outline routine in an orderly sequence.

.

Pattern Recognition Worksheet – weekly activities were organized into repeated and non-repeated activities

This worksheet follows the decomposition worksheet. Pupils identify the routines they do every day and those they do only on some days. The worksheet seeks to help pupils identify repeated routines and those that are not repeated as shown in Figure 5.

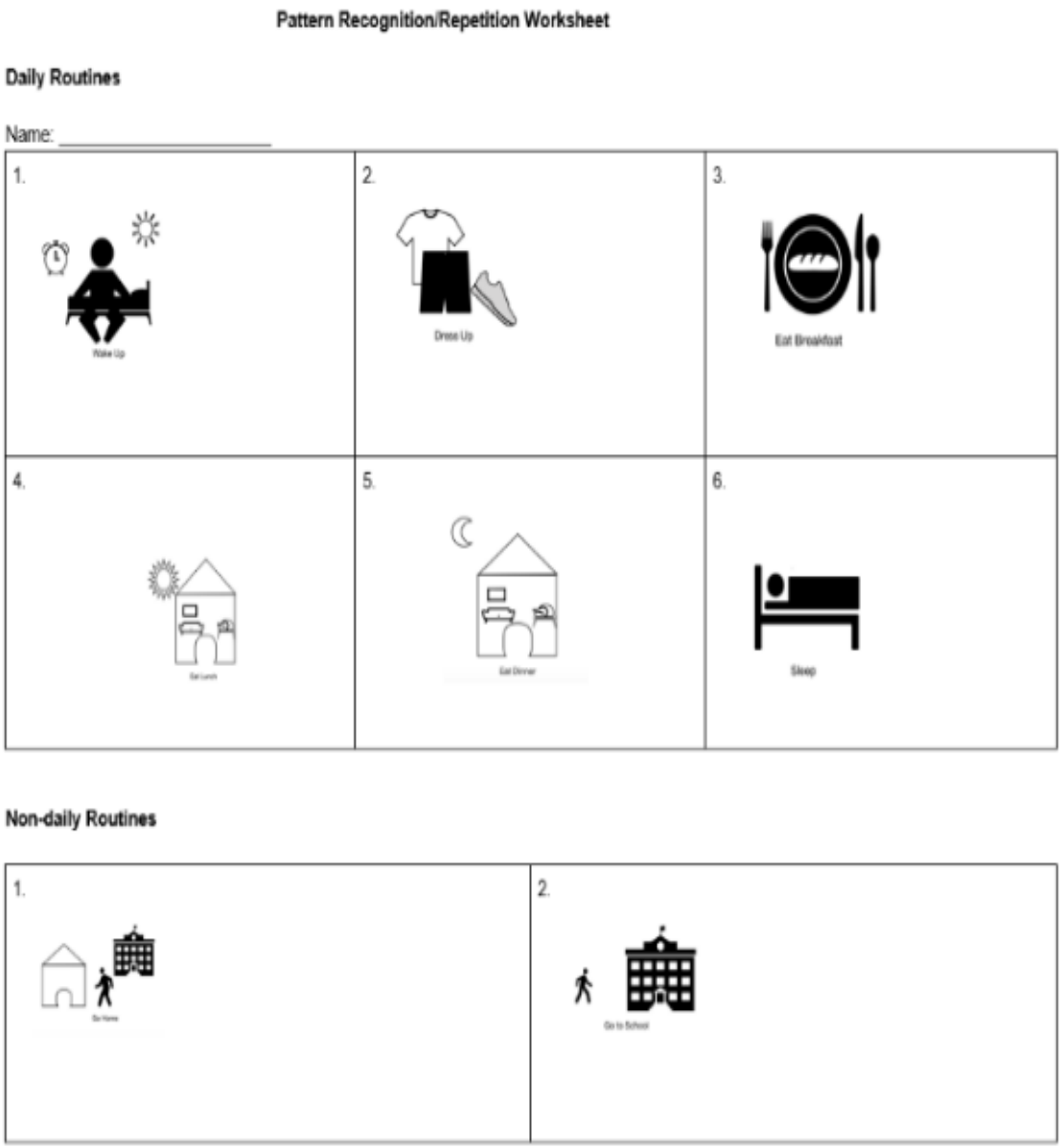
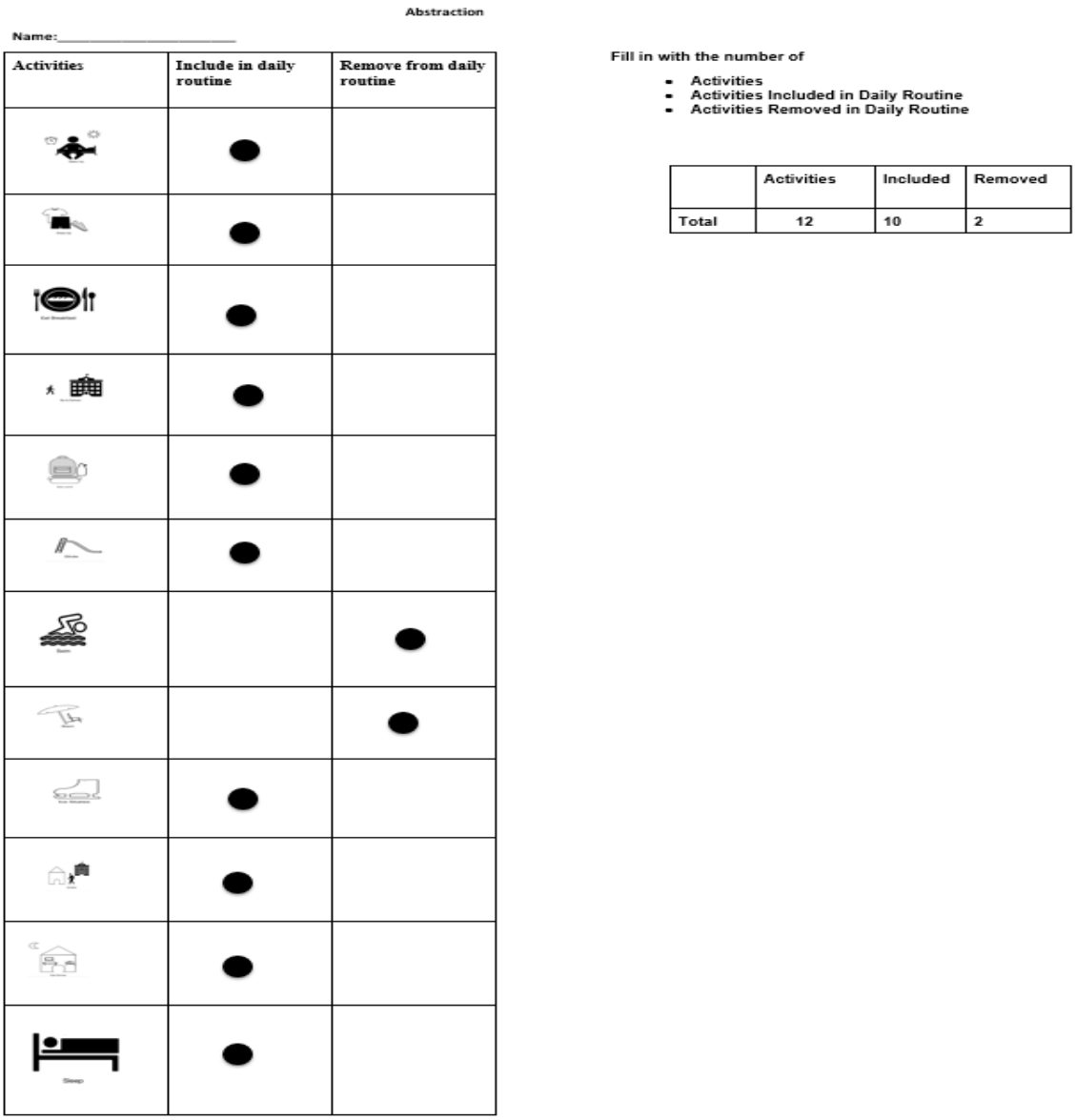


Figure 5 Pattern Recognition worksheet to organize weekly routine into repeated and non-repeated activities

Abstraction Worksheet – important weekly activities were chosen to be included into daily routines

The abstraction worksheet seeks to support the teaching and learning of the concept of algorithms through the addition or removal of certain activities from the daily routine of pupils. See Figure 6.



Abstraction

Name:_____

Activities	Include in daily routine	Remove from daily routine
	●	
	●	
	●	
	●	
	●	
	●	
		●
		●
	●	
	●	
	●	
	●	

Fill in with the number of

- Activities
- Activities Included in Daily Routine
- Activities Removed in Daily Routine

	Activities	Included	Removed
Total	12	10	2

Figure 6 Abstraction worksheet to choose important activities to include in daily routine

Decision Worksheet – deciding the action to take depending on the season (winter or summer)

This worksheet (See Figure 7) teaches decision-making using certain conditions. It is used to explain why and how we make decisions daily. Pupils state an activity they engage in depending on a condition. For example, they go skiing if it is winter and swimming if it is summer. Pupils are encouraged to come up with their own conditions and make the necessary decisions.

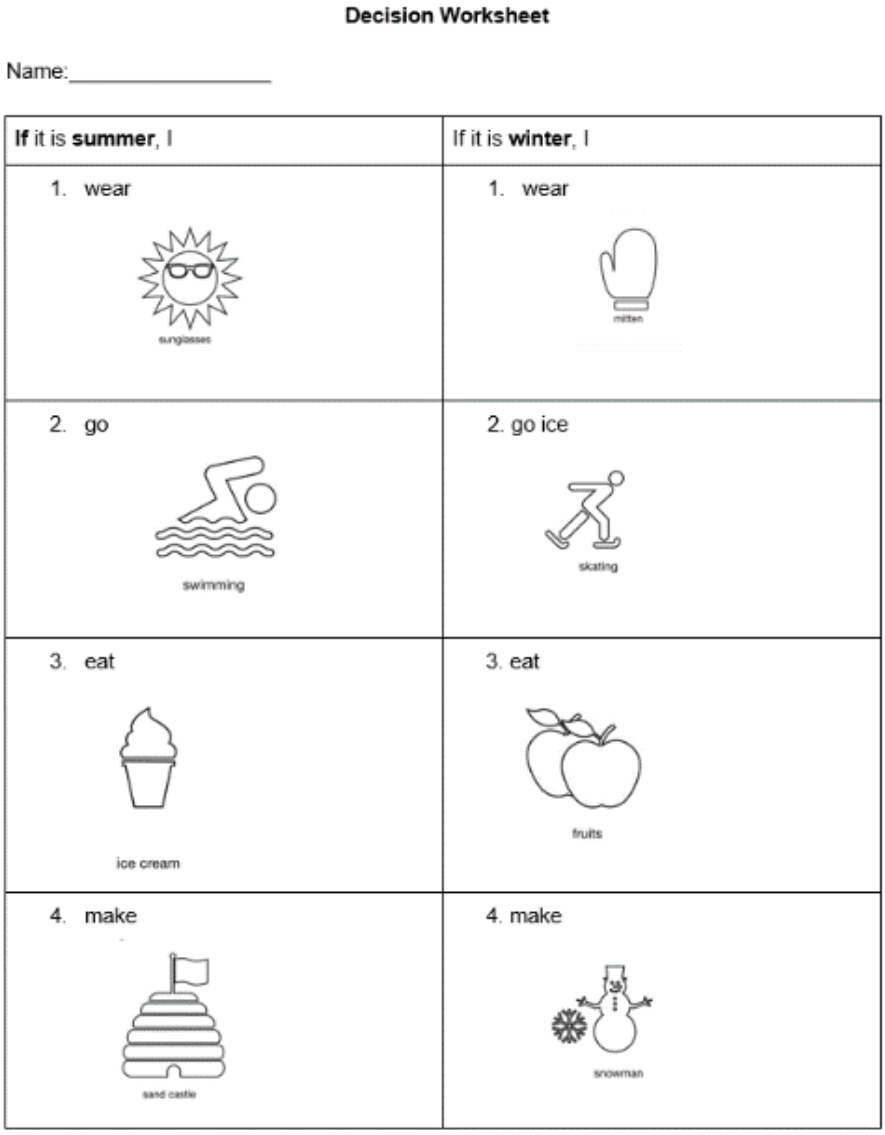


Figure 7 Decision worksheet for deciding the action to take depending on whether it is winter or summer

4.3.2 Using Scratchjr as a tool to teach computational thinking

With ScratchJr, young children (ages 5-7) can program their own interactive stories and games. In the process, they learn to solve problems, design projects, and express themselves creatively on the computer. Figure 8 shows the main interface of the tool where users can add/edit characters, add/edit scenes and create actual code. More information about Scratchjr can be found from their website (https://www.scratchjr.org/).

The example coding area (bottom of Figure 8) has two code snippets: the left code has an example where the cat will go one step forward and then jump 4+2 times if user clicks the flag (trigger). On the bottom-right side is an example of the code where code will be executed four

times (loop block) after the user has touched the cat. Inside of the loop is a code snippet to spin the cat.

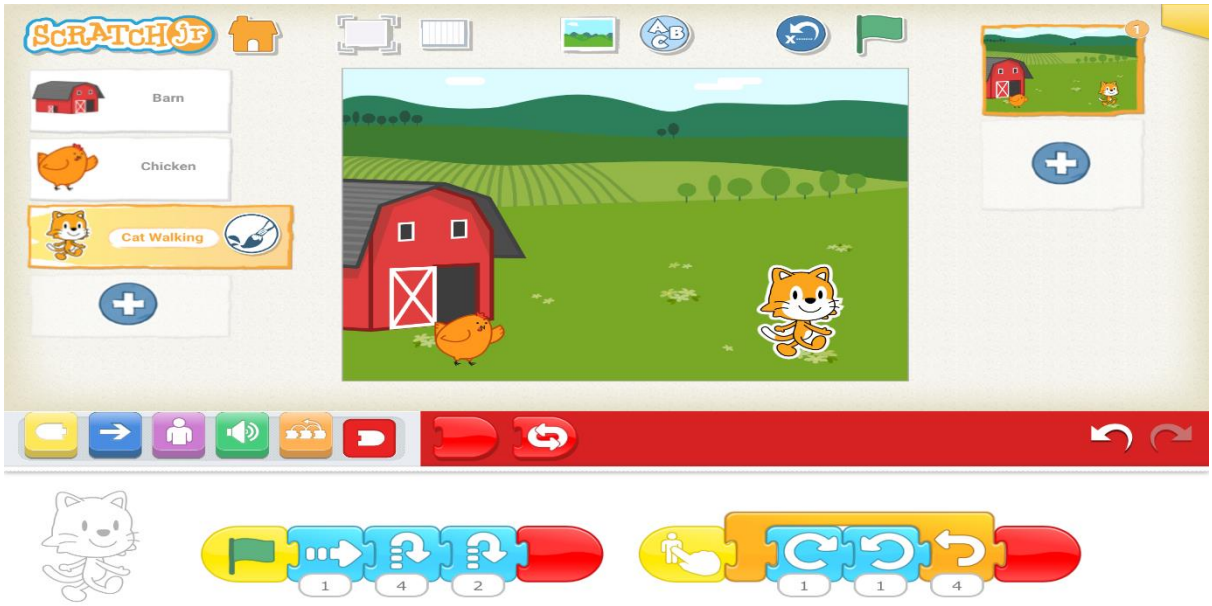


Figure 8 Scratchjr interface showing how blocks can be used to animate characters

Scratchjr was used as a programming tool in the coding club to continue with computational thinking concepts taught in the unplugged parts of the course activities (see Tables 2 and 3). Next worked examples will show how Scratchjr was used to teach computational thinking concepts.

Algorithm is a sequence of steps for doing a task

Algorithm is a sequence of steps for doing a task. Sequence is a specific order of doing something starting from the first thing to the last thing to be done. For example, an algorithm for the day can start with waking up, followed by a sequence of activities (brush your teeth, bath, eat breakfast, to the last activity, sleep). At the programming level, algorithms can be explained using start (triggering blocks), a sequence of other blocks and end with an end block (See Figure 3). In Scratchjr, the yellow blocks are triggering blocks that are used to start or trigger a character to do something. Red blocks are used to end a Scratchjr program (code). The blue blocks represent motion blocks used to move the characters around the stage. The mauve and green blocks are used to change the look of characters and add sound to code respectively.

Figure 9 illustrates how algorithms were introduced in Scratchjr. The task given was to write an algorithm to make the cat move to the position of the hen. Figure 9, presents one solution to the task. There are 4 alternative ways to make the cat move to the position of the hen using the concept of repetition. See Figure 10 (A, B, C, D).

Figure 9 Example of how algorithm and sequence can be taught in Scratchjr

Repetition – doing something over and over again.

Repetition simply means to do something over and over again. Figure 10 demonstrates 4 ways repetition was taught using Scratchjr.

1. The first way is to change the number on the motion block to 10. This reduces the number of individual motion blocks used in Figure 9 to 1 motion block as shown in Figure 10 (A)
2. The second and third way is to use the repeat block to repeat a sequence. See Figure 10 (B and C). The difference between Figure 10 (B and C) is the numbers on the blocks.
3. The fourth way is use a forever block to make the cat move to the hen repeatedly without stopping (See Figure 10 (D)).

4. In coding, it is possible to generate multiple code to accomplish the same function and must be noted that some solutions are more optimal than others. Figure 9 and Figure 10 (A, B, C, and D) demonstrates how this is possible.

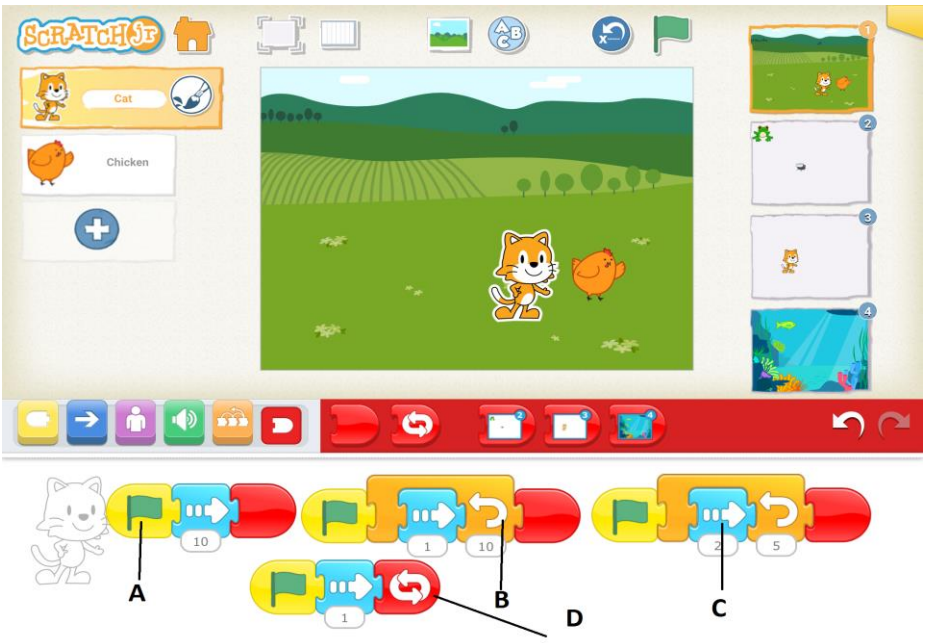


Figure 10 Multiples ways to achieve repetition in Scratchjr

Decision – making selections based on some conditions

Figure 11 Example of how to teach decisions in Scratchjr

Decision involves making a selection based on some condition. The color of message blocks is used to control the actions of the cat. Pupils had to decide what actions they wanted the cat to take. In Scratchjr, decision making can be taught using the color of the message block (See Figure 11).

Abstraction

Abstraction is about simplifying and representing concepts at a higher level by including only relevant concepts. Figure 12 illustrates how an abstraction was introduced in Scratchjr. Here, pupils had to include things that are important for their code. The relevant aspects emphasized are to include a start and end blocks in addition to others blocks that the leaners deemed important for the task given.

Figure 12 Example of how to teach abstraction in Scratchjr

4.4 Data collection

The data of this thesis consisted of the artefacts produced by pupils (unplugged coding materials i.e. task sheets and plugged coding products i.e. Scratchjr codes), notes taken by the author of the thesis and self-efficacy questionnaires concerning the coding and computational thinking concepts.

4.4.1 Collecting task sheets from unplugged coding phase and scratch code from unplugged code phase

All work products done in the unplugged and plugged coding phases were collected and analyzed. Work products were task sheets from the unplugged activities and Scratchjr programs from the respective coding lessons.

4.4.2 Lesson diary as a background material

The author of this thesis kept a lesson diary which is used as a material in this study. The author wrote short descriptions about each lesson in order to develop the course design and instructional materials, and to also help with later research reporting.

4.4.3 Self-efficacy questionnaires for measuring pupils' beliefs how they can perform CT tasks

Self-efficacy measures the innate ability of individuals to accomplish a goal or perform a task. In this study, a questionnaire which included coding and computational thinking questions was developed to explore the self-efficacy of the pupils at the end of the code club. The self-efficacy questionnaire was based on Bandura's self-efficacy scale which places emphasis on the "Can" judgement of competence (Bandura, 2005). The following are examples of the scale constructed (See Appendix I for the detailed form)

1. I can add characters to my code
2. I can code characters to move
3. I can use motion blocks to make a character move
4. I can write a story in Scratchjr

The items in the scale were presented on a 100 point scale starting from 0 (cannot do it at all) through to 20, 30 through 50 (not too sure can do it), 60 – 70 (pretty sure can do it), and 80 through 100 (certain can do it).

4.5 Data analysis

Data analyzed for this thesis includes the self-efficacy questionnaire and the work product of pupils. First, examples of Scratchjr projects of some pupils are presented and examined for CT concepts. A Mann-Whitney U test was used to analyse the self-efficacy responses of pupils to identify the differences in the abilities of pupils in different groups thus males and females, and pupils with prior and no experience in coding. The findings are presented here.

4.5.1 Using Mann-Whitney test to compare differences between girls and boys, but also non-experienced and experienced coders

The Mann-Whitney U test is a nonparametric test conducted to explore statistically significant differences between independent samples (MacFarland, & Yates, 2016). The Mann-Whitney test is appropriate for analyzing ordinal and non-normally distributed data. It compares at least two of the sample means derived from the one population and examines if the means of the sample are the same (McKnight, & Najab, 2010).

In this thesis, pupils were included to the sample based on their prior coding experience. Participating pupils were also divided by their gender (8 girls, 9 boys), so Mann-Whitney U-test was used to explore the differences between both conditions (boys vs girls and experience vs. no-experience). Those grouping conditions were used to explore differences between the means in the self-efficacy questionnaire data about coding and computational thinking.

4.5.2 Evaluating unplugged (worksheets) and plugged coding (Scratchjr code) activities

To identify and assess learning, coding schemes were developed for both unplugged (Table 5) and plugged learning activities (Table 6). Criterions used in the coding schemes were based on existing framework by Brennan & Resnick (2012). Unplugged activities were coded by analyzing how pupils had used stickers on their worksheets. For example, pattern recognition concept was understood when pupils were able to identify the activities they do daily and also activities that they do sometimes.

Plugged coding was assessed by analyzing code snippets done by the pupils. The ability to use the appropriate blocks in the right order to accomplish the task was an example of using computational thinking concept in the code.

Table 5 Coding scheme for evaluating unplugged worksheets

CT concept	Description
Algorithm	Pupils must start their code with the first thing they do in the morning to the last thing they do in the day. Pupils are expected to organize their daily activities in a sequence.
Decomposition	Pupils must organize their daily activities in order. An algorithm for each day must be created.
Pattern recognition	Based on the decomposition, pupils are to identify the activities they do every day and those they do on somedays.
Decisions	Pupils are to make decisions based on the time of the year (winter or summer). It is expected that the pupil can place the stickers at the right places.
Abstraction	Ability to select important activities or routines for the day in addition to other activities that the pupil considers important for their daily routine.

Table 6 Coding scheme for evaluating Scratchjr projects

CT concept	Coding Concept	Block	Description
Algorithms	Sequence	Start (Triggering) End (Block)	Pupil must start their code with a start block and an end block. In between these two blocks, are motion blocks.
Decomposition and Pattern recognition	Repetition	Start block Repeat block Forever block End block	Presence of start and end blocks. A sequence of motion blocks using either repeat or forever blocks.

Decision-making	Conditionals	Start block Control blocks End block (optional) Repeat block Forever block	Presence of start and end blocks. Use of control blocks for sending and receiving message using color coded start blocks.
Abstraction		Start block Look blocks Sound blocks End block Modifying characters (optional) Repeat block Forever block Message block	Presence of start and end blocks. Using sound and look blocks. Recording one's own sound Modifying characters with camera taken pic-tures
Create your Scratchjr Project (Story)	Sequence Repetition Conditionals	Start block Look blocks Sound blocks End block Modifying characters Repeat block Forever block Message block	Presence of start and end blocks. Appropriate use of blocks. (optional) Decomposition of story into scenes using pages of Scratchjr

4.5.3 Ethical issues

Permission forms were sent to parents of the participants to get their permission to collect the data of their wards and include it to research data repositories.

The data collected included worksheets, Scratchjr projects, and self-efficacy responses to questionnaire which all were used to assess participant's knowledge and ability to code. An example of a participant's work was used to illustrate how learning took place and the effectiveness of instructions. The recorded observations with notes were done by the author of the study (see Appendix II).

All the data was stored into university data storage and kept there as anonymized to ensure that the participant's identity is protected.

5 Results

5.1 How were unplugged and plugged coding activities designed in order to introduce computational thinking concepts to pupils?

Evaluation is the last phase of the ADDIE method used for this thesis (see Figure 1). It is also crucial part of the cyclical nature of the design-based research. Due to time constraints in the context of this study, only minor iteration cycles were possible to implement.

Most of the activities related to this research question are presented in the methods section in the form of pedagogical practices and instructional materials. In this section, one example of the iterations done in the context of this study is presented. Development of the unplugged instructional materials is described below.

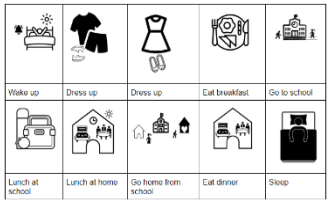
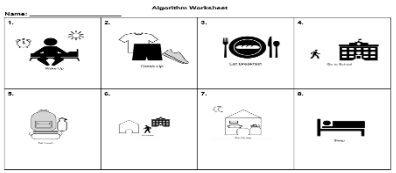
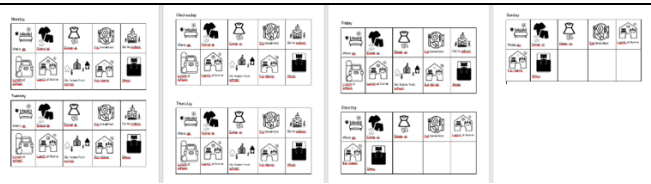
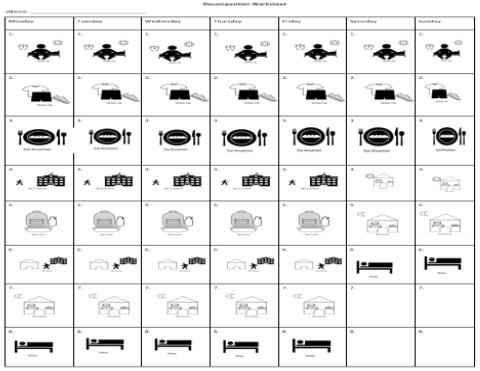
5.1.1 Prototypes with minor improvements in unplugged materials

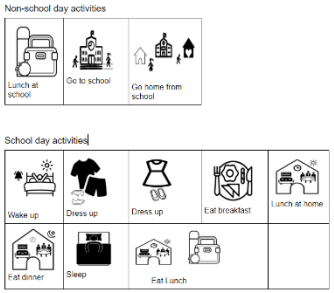
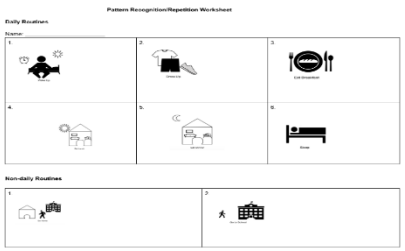
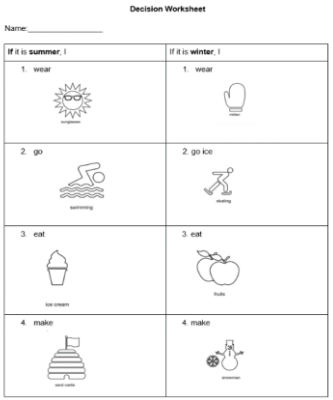
Minor changes were made to the unplugged materials before testing them at the after-school coding club, during, and after each coding club session. Changes made to the 1st prototype of the algorithm worksheet affected the 1st prototypes of the decomposition and pattern recognition worksheets. The changes included the addition of numbers to the worksheet to guide the pupils to organize their routine in an orderly fashion. The stickers for the worksheets were changed as well.

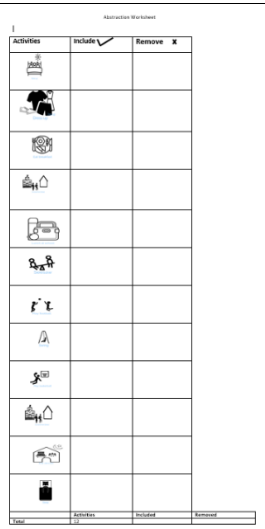
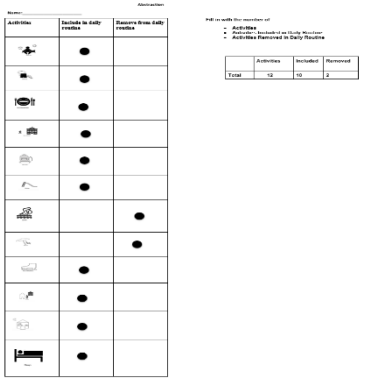
After the lesson that used the algorithm worksheet proved successful with the numbers, the decomposition and pattern recognition worksheets were redesigned during the implementation of the coding club in a similar fashion with the goal to scaffold the learning process. Changes made to the decision worksheet occurred after the implementation of the lesson. During the decision lesson, it was realized the decision pupils had to make did not match the predefined set of actions. This led to a little confusion among the pupils which was quickly clarified by explaining the concept and the idea behind the worksheet a wee bit more.

There was a need to redesign the decision worksheet. The abstraction worksheet underwent minor changes during the period of the coding club prior to the lesson on abstraction to make the worksheet friendlier and easier to use. See Table 7.

Table 7 Prototype worksheets for unplugged activities with changes made to improve worksheet

CT concept	1st Prototype	Current Prototype	Changes Made
Algorithms			Redesign of worksheet. Addition of numbers to guide pupils to organize their daily routine orderly.
Decomposition			Redesign of worksheet by organizing weekly routine in one table to make it easier to use.

Pattern recognition			Redesign of worksheet by changing the goal of the task from school and non-school day activities to daily and nondaily activities. This was done to simplify the task for pupils.
Decisions			Redesign of worksheet to include good description of the actions leaners will take when a condition is fulfilled.

Abstraction			Redesign by reorganizing the look of the worksheet to be friendlier. Text was added to make the work-sheet self-explanatory

RQ2: How were pupils' understanding of basic computational thinking concepts visible in the artefacts that they did produce during the coding club?

In this section, pupils' understanding of the basic computational thinking concepts will be presented in the form of examples from the unplugged and plugged coding tasks, as well as results of the assessment of the pupils' artefacts (activity sheets) and code (Scratchjr projects).

5.1.2 Examples of the artefacts created by pupils in the unplugged activities

In this section, examples of how pupils interacted with unplugged material are explained. There were some differences in the worksheet of pupils. See Figures 13 and 14. While some dressed up before eating breakfast, others ate breakfast before dressing up. This shows how the worksheet accommodated the routine structure of each pupil, enabling personalization. Provisions were made for gender-specific stickers for boys and girls as shown in the Figures 13 and 14.

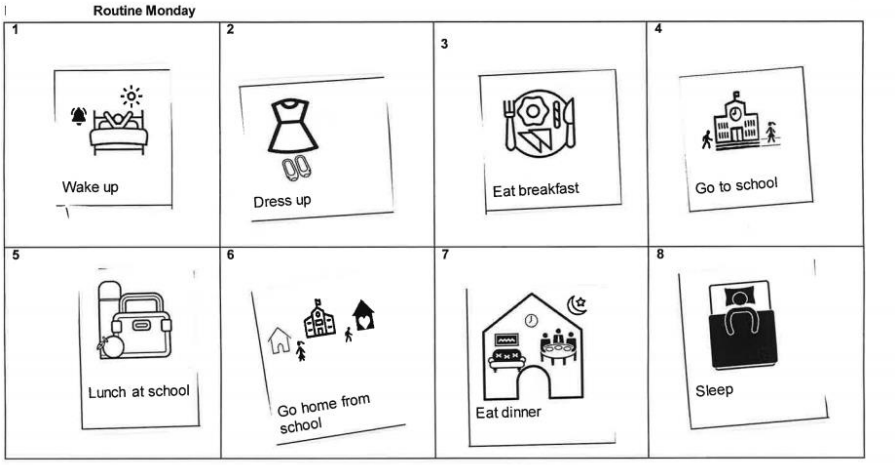


Figure 13 1st example of how pupils interacted with algorithm worksheet

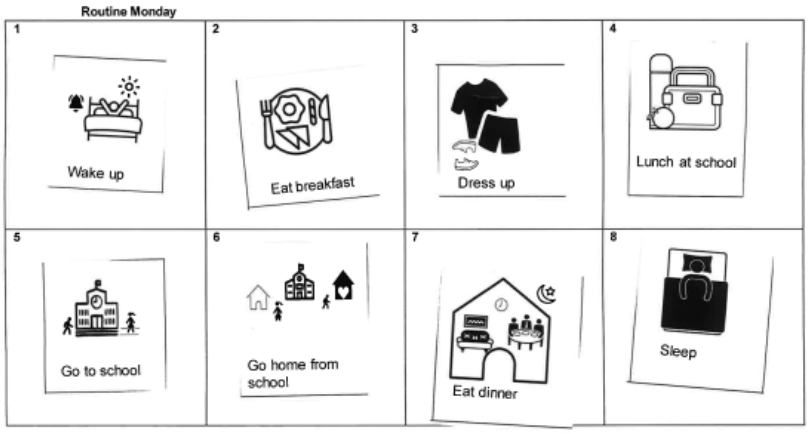


Figure 14 2nd example of how pupils interacted with algorithm worksheet

Figure 15 Example of how leaners interacted with the 1st prototype decision worksheet

5.1.3 Pupils' performance in plugged and unplugged activities

After analyzing pupils' work products thus Scratchjr projects and unplugged activity work-sheets, some pupils succeeded in performing the task given (see Tables 8 and 9). However, some were not very successful. Their work did not show evidence of the criteria described in Tables 5 and 6. It can however be concluded that many of the pupils did excel at the task given and hence the intervention was effective.

Table 8 Pupils' performance in unplugged worksheets

CT concept	No of pupils who participated	No of pupils who succeeded	No of pupils who did not succeed
Algorithm	17	17	
Decomposition	15	15	
Pattern recognition	15	15	
Decisions	12	12	
Abstraction	15	15	

Table 9 Pupils' performance in Scratchjr projects

CT concept	No of pupils whose work were saved	No of pupils who succeeded	No of pupils who did not succeed
Algorithm	13	13	
Repetition	11	10	1
Decisions	10	6	4
Abstraction	9	6	3

Table 10 Examples of Scratchjr stories

Project	Description	Teacher's Comment
	Pupil 1: Figure 11 shows an animated cat moving up and down. In this example, the pupil made use of a start and an end block to start and end code respectively. The pupil made use of two different motion blocks 3 times to make the cat move.	Teacher's comment: This pupil could have made use of a repeat block and one of each of the motion blocks instead.

	Pupil 2: Figure 12 shows an animated lizard walking on a branch of a tree. This pupil made used of the start and end block to start and end code. This pupil made use of repeat and motion blocks to sequence the movement of the lizard in a way the pupil desired.	Teacher's comment: This pupil demonstrated her ability to use the blocks in the right context.
	Pupil 3: Figure 13 shows an animated airplane flying. This pupil designed an airplane, made use of the start, repeat, motion and end blocks to animate the airplane	Teacher's comment: This pupil made use of the blocks in a way that demonstrates mastery of concepts. The pupil did a good job to design an airplane
	Pupil 4: Figure 14 shows an animated ball. The pupil chose to animate a ball. The pupil selected a space background and added a cat and an astronaut. The movement of the ball added a motion effect to this code.	Teacher's comment: This pupil demonstrated mastery by making use of the start, control blocks, motion and end blocks to create a meaningful animation.

5.2 What differences exist in the self-efficacy beliefs of learners regarding computational thinking and coding abilities with respect to prior experienced and gender?

An analysis of pupil's self-efficacy beliefs in terms of their ability to write Scratchjr story was conducted. The results revealed that 10 out of 11 pupils who participated in the survey believed they could write Scratchjr a story. This demonstrates that pupils understood the concepts such as algorithm and repetition and demonstrated this in their stories (see Table 11). The remaining 6 pupils were absent on that day.

Table 11 I can write a story in Scratchjr

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	1,00	1	5,9	9,1	9,1
	4,00	10	58,8	90,9	100,0
	Total	11	64,7	100,0	
Missing	System	6	35,3		
Total		17	100,0		

In addition, self-efficacy questions related to their ability to apply computational thinking skills in their daily lives revealed that 8 out of the 10 pupils who participated were certain of their ability to apply algorithms to their daily routines. 1 pupil was not too sure, and the remaining pupil was not sure at all that (s)he could apply algorithms in their routine (see Table 12).

Table 12 When I do my daily routines, I apply algorithms

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	10	1	5,9	10,0	10,0
	50	1	5,9	10,0	20,0
	90	2	11,8	20,0	40,0
	100	6	35,3	60,0	100,0
	Total	10	58,8	100,0	
Missing	System	7	41,2		
Total		17	100,0		

Differences in the self-efficacy beliefs of the pupils

A Mann-Whitney test conducted revealed that there were slight differences between the self-efficacy beliefs of pupils to code and perform simple computational thinking tasks though the difference is not significant. The Mann-Whitney test revealed that boys had a greater (*Mdn = 6.00*) self-efficacy belief to code a story in Scratchjr than girls (*Mdn = 5.17*), *U= 10.000*, *p=*

0.414 (see Appendices III and IV). The test showed that there was a difference in the ability of pupils to think computationally. Questions were asked related to their ability to apply CT concepts such as algorithms in their daily routine. Both boys and girls reported the same self-efficacy beliefs in terms of their ability to apply computational thinking concepts to their daily routine (When I do my routines, I apply algorithms). They had the same self-efficacy beliefs although the difference was not significant (Mdn = 5, U=12.000, p=0.881). See Appendices V and VI.

Another Mann-Whitney test conducted revealed that pupils who had prior coding experience had higher self-efficacy beliefs than pupils who had no prior experience. The test showed that pupils who had prior coding experience were more confident of their ability to write a story in Scratchjr (*Mdn = 6*) than pupils who had no prior coding experience (*Mdn = 4*) though the difference is not significant statistically (*U = 10.000, p = 0.414*). See Appendices VII and VIII. Also pupils, with prior coding experience were more confident of their ability to think computationally (When I do my routines, I apply algorithms), (*Mdn = 6*) than pupils with no such prior experience (*Mdn = 4*) although the difference is not statistically significant (*U = 8.000, p = 0.476*) See Appendices IX and X.

6 Discussion

The main purpose of this thesis is to design, and test unplugged computational activities that aim to introduce coding to pupils. To achieve this, unplugged worksheets were created for computational thinking concepts. In addition, Scratchjr lessons were designed to integrate computational thinking concepts into coding lessons.

6.1 How were unplugged and plugged coding activities designed in order to introduce computational thinking concepts to pupils?

Scratchjr, the block-based coding application supports the teaching and learning of computational thinking concepts. Computational thinking concepts such as algorithms, decomposition, pattern recognition, decision making, and abstraction can be practiced in Scratchjr.

Firstly, unplugged activities were designed for each computational thinking concept. The equivalent coding concept, if exists, was identified. The unplugged activity was used to introduce the computational thinking concept. Afterwards, its equivalent coding concept was used to reinforce the concept. Pupils had the opportunity to learn about computational thinking concepts and practice or apply them in the context of Scratchjr. Using the Scratchjr blocks such as trigger (start), end, motion, looks, sound, and control blocks, computational thinking concepts were taught to pupils. They were then given tasks to do. The solutions they produced for the tasks given had to demonstrate evidences of computational thinking concepts in their Scratchjr programs. In Scratchjr, an algorithm is equivalent to sequence.

The stories pupils created could be just a page or not more than 4 pages. For a story of one page, a sequence of blocks with starting and ending points must be present. This represented an algorithm. For stories of more than a page, the pupils had to break down the stories into scenes and identify the needed scenes such as where the story starts and where it ends as well as the scenes that come in between the start and end. Breaking down the story is known as decomposition in computational thinking. For the concept of decomposition, they had to identify patterns in their stories that repeat. The identified repeated patterns can be implemented using relevant blocks (repeat or forever block) for repeating a sequence in Scratchjr. Abstraction, another computational thinking concept was connected to Scratchjr. Abstraction was evident when pupils were able to include important scenes in their Scratchjr stories. Important scenes that were used to examine pupils' stories included the presence of starting and ending scenes. The pupils had the

opportunity to modify the characters as part of abstraction when the pupils deemed it relevant for their story. Using Scratchjr, abstract computational thinking concepts can be taught, practiced and applied in coding. Table 4 shows how unplugged activities were integrated into coding activities. Also, Table 10 shows examples of Scratchjr stories created by pupils.

6.2 How were pupils' understanding of basic computational thinking concepts visible in the artefacts that they did produce during the coding club?

During the process of creating an algorithm for their daily and weekly routines, pupils were exposed to computational thinking concepts such as algorithms, decomposition, pattern recognition, repetition and abstraction. Adapting these computational thinking concepts to coding in the context of Scratchjr gave pupils the opportunity to explore these concepts further to gain more understanding. When Scratchjr written by pupils run, they see the outcome, debug, and make the necessary changes in their program (stories). There are similar evidences that pupils learn computational thinking skills through coding (Chalmers, 2018; Marcelino, Pessoa, Vieira, Salvador, & Mendes,2018) or through unplugged activities (Curzon, 2013; Brackmann, 2017).

By evaluating pupils work products including Scratchjr projects and unplugged activities, I notice how pupils applied the concepts to write their own algorithms and animate characters in Scratchjr. See Table 10 for different stories pupils wrote to demonstrate their understanding of Scratchjr stories. Figures 13 and 14 show alternate ways pupils wrote an algorithm for their daily routines. It must be noted that pupils should be encouraged to be creative in generating solutions if the important components are present. For example, in Scratchjr, the ability of pupils to use important blocks such as the start and end blocks, together with the correct sequence of blocks served as evidence to demonstrate their understanding of the sequence, a computational thinking concept. A similar concept was used to determine if pupils understood computational thinking concepts in the unplugged lessons. Pupils demonstrated their understanding of algorithms, a CT concept when they created an algorithm for their daily routine such that it shows the first thing they do in a day (which is to wake up) and the last thing they do (which is to sleep) together with the correct sequence of other activities they do in a day.

6.3 What differences exist in the self-efficacy beliefs of learners regarding computational thinking and coding abilities with respect to prior experienced and gender?

To argue further that pupils understood the computational thinking concepts they were introduced to, frequency statistics conducted on the self-efficacy beliefs of pupils revealed that many of them were sure they could apply computational thinking concepts to their daily routines and could also write Scratchjr stories. In addition, Mann-Whitney statistics tests were conducted to identify differences between the self-efficacy beliefs of girls and that of boys as well as pupils who had prior coding experience and those who did not. The tests revealed slight differences between the self-efficacy beliefs of boys and girls as well as pupils who had prior and no coding knowledge prior to taking this computational thinking lesson.

7 Conclusion

In this research, the need for creating materials for teaching and learning computational thinking were identified. The concept of computational thinking was explored leading to the identification of key concepts such as algorithms, decomposition, pattern recognition, abstraction and decisions. The study focuses on designing teaching and learning materials for computational thinking and coding. The materials make use of real-life experiences of pupils to explain computational thinking concepts. The daily and weekly routine of pupils were used to explain CT concepts such as algorithms, decomposition, pattern recognition, and abstraction. Seasons such as winter and summer were used to explain conditionals or decision-making. Each worksheet came with a set of stickers because of the age and English-speaking ability of the pupils.

The study also explored ways in which computational thinking can aid the understanding of coding concepts. Using Scratchjr, computational thinking concepts were mapped to coding concepts. The success achieved in using this approach to explain coding concepts is attributed to the real-life experiences used to explain computational thinking concepts.

The work products pupils produced were used to evaluate their comprehension of the concepts taught. In addition, Bandura's self-efficacy questionnaire was used to examine the perceived self-efficacy beliefs of the pupils. It was identified that the pupils who had prior coding experience were more confident of their ability to think computationally and code than pupils with no prior experience though the difference was not significant. Also, while boys were more confident of their ability to code than their ability to thinking computationally, girls were more confident about their ability to think computationally. Based on these findings, coding can be introduced to novices using real-life computational thinking concepts.

CT has the potential to equip pupils with the ability to code. Computational thinking concepts can be integrated into coding lessons when teaching coding. Activities for teaching and learning can include plugged and unplugged activities. Unplugged activities provide a means to explore abstract computational thinking concepts using real-life experiences. This enhances the understanding of pupils.

The designed materials when tested during the after-school coding club proved successful after evaluating the work products of leaners. In general, the pupils were successful in the tasks they were given. They were able to apply the concepts learned in their Scratchjr projects. Also, the pupils reported positive self-efficacy beliefs which indicates that they understood the concepts

taught. Based on these evidences, it can be concluded that the designed instructional materials were effective as aids for teaching and learning computational thinking.

Conditions such as age, prior knowledge, and English-speaking ability affected the design of lesson materials. Hence future replication may require changes in the plan to accommodate the new context. This thesis serves as an example of how to design unplugged activities for teaching and learning computational thinking as well as how computational thinking concepts can be integrated into coding activities using Scratchjr.

Future research directions include conducting more tests to validate and improve unplugged instructional materials and the creation of new materials for other CT concepts.

References

Balanskat, A., & Engelhardt K. (2015). Computing our future. Computer programming and coding. Priorities, school curricula and initiatives across Europe.

Barr, D., Harrison, J., & Conery, L. (2011). Computational Thinking: A Digital Age. Retrieved from https://files.eric.ed.gov/fulltext/EJ918910.pdf

Bandura, A. (2005). Guide for constructing self-efficacy scales. In *Self-efficacy beliefs of adolescents, 5(1), 307-337.* (pp. 307–337). Retrieved from https://www.uky.edu/~eushe2/Bandura/BanduraGuide2006.pdf

Barab, S., & Squire, K. (2004). Design-based research: Putting a stake in the ground. *The journal of the learning sciences*, *13*(1), 1-14.

Bloom, B. S. (Benjamin S. (1956). *Taxonomy of educational objectives; the classification of educational goals,*. Longmans, Green.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing Computational Thinking in Compulsory Education*. https://doi.org/10.2791/792158

Brackmann, C. P., Román-González, M., Robles, G., Moreno-León, J., Casali, A., & Barone, D. (2017, November). Development of computational thinking skills through unplugged activities in primary school. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education* (pp. 65-72). ACM.

Brennan, K., Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *IN AERA 2012*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.296.6602

Chalmers, C. (2018). Robotics and computational thinking in primary school. International Journal of Child-Computer Interaction, 17, 93-100.

Clark, D. (2011). Instructional System Design: The ADDIE Model - A Handbook for Practitioners. Retrieved from http://www.nwlink.com/~donclark/hrd/sat.html

Code.org. (2013). Curriculum Guide and Unplugged Lesson Plans Computer Science Fundamentals Courses A-F. Retrieved from https://code.org/files/CSF_CoursesA-F_Curriculum_Guide.pdf

Cuny, J., Snyder, L., & Wing, J. M. (2010). *Demystifying computational thinking for non-computer scientists*.

Curzon, P., McOwan, P. W., Plant, N., & Meagher, L. R. (2014). Introducing teachers to computational thinking using unplugged storytelling. In *Proceedings of the 9th Workshop in*

Primary and Secondary Computing Education on - WiPSCE '14 (pp. 89–92). New York, New York, USA: ACM Press. https://doi.org/10.1145/2670757.2670767

Curzon, P., (2013). cs4fn and computational thinking unplugged. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education on - WiPSE '13* (pp. 47–50). New York, New York, USA: ACM Press. https://doi.org/10.1145/2532748.2611263

Design-based research. (2018, October 1). EduTech Wiki, A resource kit for educational technology teaching, practice and research. Retrieved 07:57, April 15, 2019 from http://edutechwiki.unige.ch/mediawiki/index.php?title=Design-based_research&oldid=69812.

Fuller, U., Johnson, C. G., Tuukka Ahoniemi, Kentacuk, Cukierman, D., Hernán-Losada, I., Jackova, J., … Thompson, E. (2007). Developing a Computer Science-specific Learning Taxonomy. *Developing a Computer Science-Specific Learning Taxonomy. ACM SIGCSE*, Bulletin, 39(4):152–170. Retrieved from https://www.cs.kent.ac.uk/pubs/2007/2798/content.pdf

Harlen, W., & James, M. (1997). Assessment and Learning: differences and relationships between formative and summative assessment. https://doi.org/10.1080/0969594970040304

ISTE, & CSTA. (2011). Operational Definition of Computational Thinking for k-12 Education. Retrieved from http://www.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf?sfvrsn=2

Kazimoglu, C., Kiernan, M., Bacon, L., & Mackinnon, L. (2012). A serious game for developing computational thinking and learning introductory computer programming. *Procedia-Social and Behavioural Sciences*, *47*, 1991-1999.

Lamagna, E. A. (2015). Algorithmic thinking unplugged. *Journal of Computing Sciences in Colleges*, *30*(6), 45–52. Retrieved from https://dl.acm.org/citation.cfm?id=2753036

Lambert, L. (2009). Computer science outreach in an elementary school. *Journal of Computing Sciences in Colleges*, *24*(3), 118–124. Retrieved from https://dl.acm.org/citation.cfm?id=1409896

Lave, J. (1988). Cognition in practice. *Applied Cognitive Psychology*, *4*(6), 504–506. https://doi.org/10.1002/acp.2350040610

LeMay, S., Costantino, T., O'Connor, S., & ContePitcher, E. (2014). Screen time for children. In *Proceedings of the 2014 conference on Interaction design and children - IDC '14* (pp. 217–220). New York, New York, USA: ACM Press. https://doi.org/10.1145/2593968.2610456

Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M., & Rusk, N. (2008). *Programming by choice: urban youth learning programming with scratch* (Vol. 40, No. 1, pp. 367-371). ACM

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014, June). Computational thinking in K-9 education. In *Proceedings of the working group reports of the 2014 on innovation & technology in computer science education conference* (pp. 1-29). ACM.

Marcelino, M. J., Pessoa, T., Vieira, C., Salvador, T., & Mendes, A. J. (2018). Learning Computational Thinking and scratch at distance. *Computers in Human Behavior*, *80*(C), 470-477.

MacFarland, T. W., & Yates, J. M. (2016). Mann–whitney u test. In *Introduction to nonparametric statistics for the biological sciences using R* (pp. 103-132). Springer, Cham.

Namukasa, I. K., Kotsopoulos, D., Floyd, L., Weber, J., Kafai, Y., Khan, S. Somanath, S. (n.d.). From computational thinking to computational participation: Towards Achieving Excellence through Coding in elementary schools. Retrieved from http://researchideas.ca/coding/docs/CT-participation.pdf

Nishida, T., Kanemune, S., Idosaka, Y., Namiki, M., Bell, T., & Kuno, Y. (2009). A CS Unplugged Design Pattern. Retrieved from http://www.computacional.com.br/arquivos/Artigos CS Unplugged - Desplugado/NISHIDA - A CS Unplugged Design Pattern.pdf

O'Neill, G., & Murphy, F. (2010). UCD TEACHING AND LEARNING/ RESOURCES Guide to Taxonomies of Learning ASSESSMENT. Retrieved from http://www.ucd.ie/t4cms/ucdtla0034.pdf

Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. New York: Basic Books, Inc. Retrieved from http://worrydream.com/refs/Papert - Mindstorms 1st ed.pdf

Resnick, M., Maloney, Andrés Monroy-Hernández, J., Rusk, N., Eastmond, E., Brennan, K., Rosenbaum, E., … Kafai, Y. (2009). Digital fluency Sould mean designing, creating, and remixing, not just browsing, chatting, and interacting. *Communications of the ACM*, *52*(11). https://doi.org/10.1145/1592761.1592779

Reigeluth, C. M. (Ed.). (2013). Instructional design theories and models: An overview of their current status. Routledge.

Starr, C. W., Manaris, B., & Stalvey, R. H. (2008). Bloom's Taxonomy Revisited: Specifying Assessable Learning Objectives in Computer Science. *ACM SIGCSE' 08* . Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.446.342&rep=rep1&type=pdf

Tedre, M., & Denning, P. J. (2016). The Long Quest for Computational Thinking. In *Koli Calling Conference on Computing Education Research* (pp. 120–129). https://doi.org/10.1145/2999541.2999542

Thies, R., & Vahrenhold, J. (2012). Reflections on outreach programs in CS classes. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education - SIGCSE '12* (p. 487). New York, New York, USA: ACM Press. https://doi.org/10.1145/2157136.2157281

Thies, R., & Vahrenhold, J. (2013). On plugging "unplugged" into CS classes. In *Proceeding of the 44th ACM technical symposium on Computer science education - SIGCSE '13* (p. 365). New York, New York, USA: ACM Press. https://doi.org/10.1145/2445196.2445303

Vygotsky, L. (1986). *Thought and Language*. Retrieved from http://s-f-walker.org.uk/pubsebooks/pdfs/Vygotsky_Thought_and_Language.pdf

Wang, M., & Wang, Y. F. (2016). A Study on Computer Teaching Based on Computational Thinking. *International Journal of Emerging Technologies in Learning (IJET)*, *11*(12), 72. https://doi.org/10.3991/ijet.v11i12.6069

Wing, J. (2006). *Computational Thinking*. Retrieved from https://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf

Wing, J. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *366*(1881), 3717–3725. https://doi.org/10.1098/rsta.2008.0118

Yadav, A., David Burkhart, Daniel Moix, Eric Snow, Padmaja Bandaru, & Lissa Clayborn. (2015). Sowing the Seeds: A Landscape Study on Assessment in Secondary Computer Science Education. Retrieved from https://www.researchgate.net/publication/281792891

Yang, F., & Dong, Z. (2017). Educational Theory (pp. 15–29). https://doi.org/10.1007/978-981-10-1944-9_2

Appendix I

Name: _____

Computer programming and Computational Thinking Self-Efficacy Scale

Rate your degree of confidence by recording a number from 0 to 100 using the scale given below:

0	10	20	30	40	50	60	70	80	90	100
Cannot do at all			Not too sure can do it			Pretty sure can do it		Certain can do it		

Self-efficacy in Coding	**Confidence (0 - 100)**
1. I can add characters to my code	_____
2. I can code characters to move	_____
3. I can use motion blocks to make a character move	_____
4. I can repeat a character's movement	_____
5. I can use repeat block in my code	_____
6. I can use forever block in my code	_____
7. I can use the message block	_____
8. I can start and end my code	_____
9. I can change the color of a character	_____
10. I can add pictures to my character	_____
11. I can add sound to my code	_____
12. I can add text to my code	_____
13. I can write a sequence for my character to move	_____
14. I can write a story in Scratchjr	_____

Self-efficacy in Computational Thinking	
1. When I do my daily routines, I apply algorithms	_____
2. When I do something often, I repeat	_____
3. I make decisions everyday	_____
4. I can breakdown a big task into smaller tasks	_____
5. I know important things I have to do everyday	_____
6. I know things I do not consider as important	_____
7. I can identify things that are similar	_____
8. I can identify things that are different	_____

Appendix II

Researcher's notes

29th October, 2018

The original plan was to have an unplugged lesson, followed by a short lecture, 2-3 worked examples and a collaborative task.

The unplugged lesson was about binary coding. The pupils had to write their names in binary code. The purpose of this lesson was to give an idea of the language of the computer and how everything that runs on the computer is written in computer language.

This was followed by introducing them to the Scratchjr app interface. After seeing how the pupils immerse themselves with Scratchjr, I decided to give them the opportunity to explore the interface more by themselves after teaching the main concept (coding).

I had a parent who was very helpful in delivering the lesson and supporting the lesson plan.

5th November 2018

The lesson began with a short introduction of the concept of sequence and algorithm, emphasizing on the starting point and the ending point. Afterwards, the pupils did an unplugged activity on their daily routine. The purpose was to reinforce the idea of sequence, algorithm. A daily routine began by waking up followed by a series of activities and ends by sleeping. Most of the sequence of activities were similar for some pupils except for a few in which two activities swapped places. In all, the pupils understood the concept. The two concepts were later used in Scratch. The pupils were given two worked examples that used the start and end blocks to move characters using the motion blocks. Afterwards, the pupils were given the opportunity to explore Scratch themselves using the two concepts. With the help of the co-teacher, I went around to check their work and encouraged them to add an end block which was missing in most of their exploratory work. This provided another opportunity to emphasize the relevance of the concepts.

12th November 2018

The lesson began with a short introduction of the concept of repetition. We recapped the algorithm lesson because I wanted to reinforce the concept again. I asked a question relating to repetition which a pupil responded. The pupils received an unplugged activity relating to repetition. The activity built on the previous activity. This time around, pupils built a storyboard that contained activities for each day of the week. A group of 4 pupils were given a storyboard sheet to fill in their daily activities based on the day. The original lesson plan included two unplugged activities for repetition and pattern recognition. The idea of the pattern recognition was to identify activities that occur every day as daily activities and those that occur on some days as nondaily activities from the repetition worksheet. These activities could also be described as school day verses non-school day activities. However, this unplugged activity was not done because of the time factor. After the unplugged, we revisited the algorithm concept in Scratchjr. Using, the same example (move towards an object with one motion block), the sequences of blocks were minimized using the number to repeat a block, using the repeat block, and forever loop. I demonstrated how the repeat block and motion block with numbers could be used to achieve the same purpose in this particular example. In this lesson, the co-teacher and I went around to assist pupils who needed help to understand and repeat the worked example.

19th November 2018

This lesson began with a short introduction to the concept of conditionals. The relevance of it for making decisions was highlighted. The unplugged activity had a set of pictures for summer and winter clothes and activities and a worksheet. The pupils were to put a picture of activity or clothes based on the season. The worksheet began with “if it is ____”

After the pupils had finished the unplugged activity, we proceeded to Scratchjr. Here, a worked example of conditionals was shown using the send and receive message block. A character performed an action when it received a message which had been colour coded. It was very tough to explain the conditionals concept given that Scratchjr does not have this structure. After multiple explanations, the pupils got the concept. Pupils were encouraged to use the previous concepts taught in their worked examples.

26th November 2018

This lesson also began with an explanation of the concept of abstraction. How to include or remove certain things from your code. The lesson started with an unplugged activity. The unplugged activity included a list of daily activities and extra activities. The pupils were asked to add important routines and remove unimportant routines. This was followed by a plugged activity. The pupils were given a worked example that demonstrated important parts of programming (including a start, a sequence of blocks, and an end). In addition, the pupils were introduced to look and sound blocks. The pupils also had the opportunity to include their own pictures to their characters.

3rd December 2018

The main concepts of programming have been taught. This is a preparatory lesson for the pupils' projects. A story prompt was given to the pupils to write a story. The idea is to assess pupil's ability to use the blocks already learned to create a story. This lesson was loosely scripted but had prompts to assist the pupils to write their story. Also as the teacher, I went around to help any pupil who needed further assistance (scaffolding). I think the lesson was successful as the pupils were able to use the concepts learned in various ways. This shows that the pupils understood the concepts that have been taught. This project was intended to be a preparatory lesson, but it turned out differently hence there was no need to ask the pupils to write another story.

10th December 2018

Based on the outcome of the previous lesson, I decided to take a different approach to today's lesson. I used the Solve-It assessment from Scratchjr website. Pupils were asked to write the code after been shown a sequence of movement. Most of the pupils were able to write the correct code, others were close which is very good because there are multiple ways to achieve the same thing in coding though some code snippets are more efficient than others. However, others needed some hints to write the code successfully. The hints were using printed coding blocks.

17th December 2018

Today, the pupils created their own projects, exploring Scratchjr. They were also given efficacy questionnaire to assess their beliefs regarding their understanding of CT and coding concepts. Many of the pupils were absent, a challenge faced almost every week.

Appendix III

Ranks

Gender		N	Mean Rank	Sum of Ranks
I can add characters to my code	female	6	5,58	33,50
	male	5	6,50	32,50
	Total	11		
I can code characters to move	female	6	5,58	33,50
	male	5	6,50	32,50
	Total	11		
I can use motion blocks to make a character	female	6	6,00	36,00
	male	5	6,00	30,00
	Total	11		
I can repeat a character's movement	female	6	6,67	40,00
	male	5	5,20	26,00
	Total	11		
I can use repeat block in my code	female	6	6,00	36,00
	male	5	6,00	30,00
	Total	11		
I can use forever block in my code	female	6	5,50	33,00
	male	5	6,60	33,00
	Total	11		
I can use the message block	female	6	6,00	36,00
	male	5	6,00	30,00
	Total	11		
I can start and end my code	female	6	5,58	33,50
	male	5	6,50	32,50
	Total	11		
I can change the color of a character	female	6	5,50	33,00
	male	5	6,60	33,00
	Total	11		
I can add pictures to my character	female	6	6,08	36,50
	male	5	5,90	29,50
	Total	11		
I can add sound to my code	female	6	6,17	37,00
	male	5	5,80	29,00
	Total	11		
I can add text to my code	female	6	5,50	33,00
	male	4	5,50	22,00
	Total	10		
I can write a sequence for my character to move	female	6	5,58	33,50
	male	4	5,38	21,50
	Total	10		
I can write a story in Scratchjr	female	6	5,17	31,00
	male	4	6,00	24,00
	Total	10		

Appendix IV

Test Statistics[a]

Dependent Variables	Mann-Whitney U	Wilcoxon W	Z	Asymp. Sig. (2-tailed)
I can add characters to my code	12,500	33,500	-0,913	0,361
I can code characters to move	12,500	33,500	-0,913	0,361
I can use motion blocks to make a character move	15,000	30,000	0,000	1,000
I can repeat a character's movement	11,000	26,000	-0,932	0,351
I can use repeat block in my code	15,000	30,000	0,000	1,000
I can use forever block in my code	12,000	33,000	-0,699	0,484
I can use the message block	15,000	30,000	0,000	1,000
I can start and end my code	12,500	33,500	-0,913	0,361
I can change the color of a character	12,000	33,000	-0,697	0,486
I can add pictures to my character	14,500	29,500	-0,136	0,892
I can add sound to my code	14,000	29,000	-0,271	0,787
I can add text to my code	12,000	22,000	0,000	1,000
I can write a sequence for my character to move	11,500	21,500	-0,152	0,879
I can write a story in Scratchjr	10,000	31,000	-0,816	0,414

a. Grouping Variable: Gender

b. Not corrected for ties.

Appendix V

Ranks

Gender		N	Mean Rank	Sum of Ranks
When I do my daily routines, I apply algorithms	female	5	5,60	28,00
	male	5	5,40	27,00
	Total	10		
When I do something often, I repeat	female	5	5,20	26,00
	male	4	4,75	19,00
	Total	9		
I make decisions everyday	female	5	5,00	25,00
	male	5	6,00	30,00
	Total	10		
I can breakdown a big task into smaller tasks	female	5	4,50	22,50
	male	5	6,50	32,50
	Total	10		
I know important things I have to do everyday	female	5	5,10	25,50
	male	4	4,88	19,50
	Total	9		
I know things I do not consider as important	female	5	4,20	21,00
	male	4	6,00	24,00
	Total	9		
I can identify things that are similar	female	5	4,90	24,50
	male	5	6,10	30,50
	Total	10		
I can identify things that are different	female	5	4,70	23,50
	male	5	6,30	31,50
	Total	10		

Appendix VI

Test Statistics[a]

	Mann-Whitney U	Wilcoxon W	Z	Asymp. Sig. (2-tailed)	Exact Sig. [2*(1-tailed Sig.)]
When I do my daily routines, I apply algorithms	12,000	27,000	-0,149	0,881	1,000[b]
When I do something often, I repeat	9,000	19,000	-0,335	0,737	,905[b]
I make decisions everyday	10,000	25,000	-1,000	0,317	,690[b]
I can breakdown a big task into smaller tasks	7,500	22,500	-1,491	0,136	,310[b]
I know important things I have to do everyday	9,500	19,500	-0,169	0,866	,905[b]
I know things I do not consider as important	6,000	21,000	-1,352	0,176	,413[b]
I can identify things that are similar	9,500	24,500	-0,680	0,496	,548[b]
I can identify things that are different	8,500	23,500	-0,945	0,345	,421[b]

a. Grouping Variable: Gender

b. Not corrected for ties.

Appendix VII

Ranks

Have you used Scratchjr before?		N	Mean Rank	Sum of Ranks
I can add characters to my code	Yes	6	5,58	33,50
	No	5	6,50	32,50
	Total	11		
I can code characters to move	Yes	6	5,58	33,50
	No	5	6,50	32,50
	Total	11		
I can use motion blocks to make a character move	Yes	6	5,17	31,00
	No	5	7,00	35,00
	Total	11		
I can repeat a character's movement	Yes	6	5,58	33,50
	No	5	6,50	32,50
	Total	11		
I can use repeat block in my code	Yes	6	6,00	36,00
	No	5	6,00	30,00
	Total	11		
I can use forever block in my code	Yes	6	5,50	33,00
	No	5	6,60	33,00
	Total	11		
I can use the message block	Yes	6	6,00	36,00
	No	5	6,00	30,00
	Total	11		
I can start and end my code	Yes	6	5,58	33,50
	No	5	6,50	32,50
	Total	11		
I can change the color of a character	Yes	6	5,83	35,00
	No	5	6,20	31,00
	Total	11		
I can add pictures to my character	Yes	6	5,17	31,00
	No	5	7,00	35,00
	Total	11		
I can add sound to my code	Yes	6	5,17	31,00
	No	5	7,00	35,00
	Total	11		
I can add text to my code	Yes	6	5,50	33,00
	No	4	5,50	22,00
	Total	10		
I can write a sequence for my character to move	Yes	6	4,83	29,00
	No	4	6,50	26,00
	Total	10		
I can write a story in Scratchjr	Yes	6	5,17	31,00
	No	4	6,00	24,00
	Total	10		

Appendix VIII

Test Statistics [a]

	Mann-Whitney U	Wilcoxon W	Z	Asymp. Sig. (2-tailed)	Exact Sig. [2*(1-tailed Sig.)]
I can add characters to my code	12,500	33,500	-0,913	0,361	,662[b]
I can code characters to move	12,500	33,500	-0,913	0,361	,662[b]
I can use motion blocks to make a character move	10,000	31,000	-1,354	0,176	,429[b]
I can repeat a character's movement	12,500	33,500	-0,583	0,560	,662[b]
I can use repeat block in my code	15,000	30,000	0,000	1,000	1,000[b]
I can use forever block in my code	12,000	33,000	-0,699	0,484	,662[b]
I can use the message block	15,000	30,000	0,000	1,000	1,000[b]
I can start and end my code	12,500	33,500	-0,913	0,361	,662[b]
I can change the color of a character	14,000	35,000	-0,232	0,816	,931[b]
I can add pictures to my character	10,000	31,000	-1,361	0,174	,429[b]
I can add sound to my code	10,000	31,000	-1,354	0,176	,429[b]
I can add text to my code	12,000	22,000	0,000	1,000	1,000[b]
I can write a sequence for my character to move	8,000	29,000	-1,217	0,224	,476[b]
I can write a story in Scratchjr	10,000	31,000	-0,816	0,414	,762[b]

a. Grouping Variable: Have you used Scratchjr before?

b. Not corrected for ties.

Appendix IX

Ranks

Have you used Scratchjr before?		N	Mean Rank	Sum of Ranks
When I do my daily routines, I apply algorithms	Yes	6	4,83	29,00
	No	4	6,50	26,00
	Total	10		
When I do something often, I repeat	Yes	6	4,50	27,00
	No	3	6,00	18,00
	Total	9		
I make decisions everyday	Yes	6	5,17	31,00
	No	4	6,00	24,00
	Total	10		
I can breakdown a big task into smaller tasks	Yes	6	5,58	33,50
	No	4	5,38	21,50
	Total	10		
I know important things I have to do everyday	Yes	6	4,50	27,00
	No	3	6,00	18,00
	Total	9		
I know things I do not consider as important	Yes	6	5,25	31,50
	No	3	4,50	13,50
	Total	9		
I can identify things that are similar	Yes	6	6,00	36,00
	No	4	4,75	19,00
	Total	10		
I can identify things that are different	Yes	6	5,50	33,00
	No	4	5,50	22,00
	Total	10		

Appendix X

Test Statistics[a]

	Mann-Whitney U	Wilcoxon W	Z	Asymp. Sig. (2-tailed)	Exact Sig. [2*(1-tailed Sig.)]
When I do my daily routines, I apply algorithms	8,000	29,000	-1,217	0,224	,476[b]
When I do something often, I repeat	6,000	27,000	-1,061	0,289	,548[b]
I make decisions everyday	10,000	31,000	-0,816	0,414	,762[b]
I can breakdown a big task into smaller tasks	11,500	21,500	-0,152	0,879	,914[b]
I know important things I have to do everyday	6,000	27,000	-1,069	0,285	,548[b]
I know things I do not consider as important	7,500	13,500	-0,535	0,593	,714[b]
I can identify things that are similar	9,000	19,000	-0,694	0,487	,610[b]
I can identify things that are different	12,000	22,000	0,000	1,000	1,000[b]

a. Grouping Variable: Have you used Scratchjr before?

b. Not corrected for ties.